

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Б1.В.09

(индекс дисциплины)

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Технологии и средства конструирования программного обеспечения
(наименование дисциплины)

по направлению подготовки

09.03.03 Прикладная информатика

направленность (профиль)
Автоматизация бизнес-процессов и проектирование ИТ-решений

Форма обучения: заочная

Год набора: 2024

Общая трудоемкость: **4 ЗЕ**

Распределение часов дисциплины по семестрам

Семестр		3	Итого
Вид занятий	Форма контроля	экзамен	
Лекции		4	4
Лабораторные			
Практические			
Руководство: курсовые работы (проекты) / РГР			
Промежуточная аттестация		0,35	0,35
Контактная работа		4,35	4,35
Самостоятельная работа		131	131
Контроль		8,65	8,65
Итого		144	144

Рабочую программу составил(и):

Доцент института цифровых технологий доцент к.п.н. Копша О.Ю.

(должность, ученое звание, степень, Фамилия И.О.)

Рецензирование рабочей программы дисциплины:



Отсутствует



Рецензент

(должность, ученое звание, степень, Фамилия И.О.)

Рабочая программа составлена на основании ФГОС ВО и учебного плана направления подготовки

09.03.03 Прикладная информатика

(код и наименование направления подготовки, специальности в соответствии с ФГОС ВПО)

Срок действия рабочей программы дисциплины до «31» августа 2031 г.

УТВЕРЖДЕНО

На заседании института цифровых технологий

(протокол заседания № 1 от «5» сентября 2025 г.).

1. Цель освоения дисциплины

Цель – изучение студентами основ разработки программного обеспечения, моделей и языков конструирования, современным технологиям в конструировании программного обеспечения, инструментами, используемыми для разработки программного обеспечения, основам тестирования и сопровождения программного обеспечения.

Задачи:

1. Дать основы управления разработкой (конструированием) программного обеспечения.
2. Дать основные понятия и определения в области разработки (конструирования) программного обеспечения.
3. Дать и получить навыки по современным технологиям конструирования программного обеспечения.
4. Дать и получить навыки по практической реализации процессов конструирования.
5. Дать и получить навыки по практической работе в современных инструментах конструирования программного обеспечения.

2. Место дисциплины (учебного курса) в структуре ОПОП ВО

Данная дисциплина (учебный курс) относится к Б1 "Дисциплины (модули)" (Вариативная часть).

Дисциплины, учебные курсы, на освоении которых базируется данная дисциплина (учебный курс).

Дисциплины, учебные курсы, для которых необходимы знания, умения, навыки, приобретаемые в результате изучения данной дисциплины (учебного курса).

3. Планируемые результаты обучения

Формируемые и контролируемые компетенции (код и наименование)	Индикаторы достижения компетенций (код и наименование)	Планируемые результаты обучения
ПК-3 Способен проектировать информационные системы по видам обеспечения	ПК-3.1 Знает технологии проектирования информационных систем	Знать: технологии проектирования информационных систем Уметь: применять технологии проектирования информационных систем Владеть: навыками проектирования информационных систем
	ПК-3.2 Умеет проектировать информационные системы по видам обеспечения	Знать: виды обеспечения информационных систем Уметь: проектировать информационные системы по видам обеспечения Владеть: методами проектирования информационных систем по видам обеспечения

	<p>ПК-3.3 Владеет навыками проектирования информационных систем современными инструментальными средствами</p>	<p>Знать: технологии разработки программного обеспечения на современных языках программирования, методы адаптации прикладного программного обеспечения</p> <p>Уметь: разрабатывать программное обеспечение на современных языках программирования, применять методы адаптации прикладного программного обеспечения</p> <p>Владеть: навыками разработки программного обеспечения на современных языках программирования и методами его адаптации</p>
--	---	---

4. Структура и содержание дисциплины

Модуль (раздел)	Вид учебной работы	Наименование тем занятий (учебной работы)	Семестр	Объем, ч.	Баллы	Интерактив, ч.	Формы текущего контроля (наименование оценоч-
1. Основы конструирования программного обеспечения	Лек.	Тема 1.1. Фундаментальные основы конструирования программного обеспечения	3	2		-	
	Ср.	Тема 1.2. Стандарты в конструировании	3	14		-	
2. Инструменты конструирования программного обеспечения	Лек.	Тема 2.1. Инструменты разработки программного обеспечения	3	2		-	
	Ср.	Установка и настройка среды разработки	3	14	16	-	Отчет по практической работе
3. Управление конструированием программным обеспечением	Ср.	Тема 3.1. Управление конструированием программным обеспечением	3	14		-	
	Ср.	Тема 3.2. Основы практической реализации процессов конструирования	3	14		-	
	Ср.	Создание простейшей Web страницы	3	15	16	-	Отчет по практической работе
	Ср.	Тема 4.1. Шаблоны в программировании. Основы реализации отказоустойчивости при программировании	3	14		-	
	Ср.	Создание простейшего сервлета	3	16	16	-	Отчет по практической работе
	Ср.	Тема 4.2. Современные технологии конструирования программного обеспечения	3	14		-	

	Ср.	Разработка Web приложения с использованием технологии EJB и JPA	3	16	12	-	Отчет по практической работе
	ПА	Промежуточная аттестация	3	0,35		-	Итоговый тест по курсу через ОТ
	Контроль	Экзамен	3	8,65	40	-	
Итого				144	100		

Схема расчета итогового балла: текущий рейтинг (все занятия и промежуточные тесты) + Результат итогового теста

5. Образовательные технологии

При изучении дисциплины (учебного курса) используются дистанционные образовательные технологии.

6. Методические указания по освоению дисциплины

6.1 Рекомендации по подготовке к практическим занятиям

Обучающимся следует при подготовке к практическим занятиям следует обязательно использовать не только лекции, учебную литературу, но и другие источники;

Для того чтобы практические занятия приносили максимальную пользу, необходимо помнить, что решение задач проводятся по рассмотренному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться обучающимся на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях обучающийся не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения задачи, то нужно сравнить их и выбрать самый рациональный. Полезно до начала решения задачи составить краткий план решения задачи. Решение проблемных задач или примеров следует излагать подробно, отделяя вспомогательные пути решения от основных. Решения при необходимости нужно сопровождать комментариями, схемами, алгоритмами.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

6.2 Рекомендации по подготовке к тестированию по темам курса

Тесты – это вопросы или задания, предусматривающие конкретный, краткий, четкий ответ на имеющиеся эталоны ответов.

При самостоятельной подготовке к тестированию студенту необходимо:

а) готовясь к тестированию, проработайте информационный материал по дисциплине. Проконсультируйтесь с преподавателем по вопросу выбора учебной литературы;

б) четко выясните все условия тестирования заранее. Вы должны знать, сколько тестов Вам будет предложено, сколько времени отводится на тестирование, какова система оценки результатов и т.д.;

в) приступая к работе с тестами, внимательно и до конца прочтите вопрос и предлагаемые варианты ответов. Выберите правильные (их может быть несколько). На отдельном листке ответов выпишите цифру вопроса и буквы, соответствующие правильным ответам;

г) в процессе решения желательно применять несколько подходов в решении задания. Это позволяет максимально гибко оперировать методами решения, находя каждый раз оптимальный вариант.

д) если Вы встретили чрезвычайно трудный для Вас вопрос, не тратьте много времени на него. Переходите к другим тестам. Вернитесь к трудному вопросу в конце.

е) обязательно оставьте время для проверки ответов, чтобы избежать механических ошибок.

Тестирование позволяет оценить знание фактического материала, умение логически мыслить, способность к рефлексии и творчески подходить к решению поставленной задачи

6.3. Рекомендации по подготовке к итоговой сдаче дисциплины

Подготовка к итоговой сдаче предмета способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к ней, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На итоговой сдаче студент демонстрирует то, что он приобрел в процессе обучения по конкретной учебной дисциплине.

Необходимо ориентировать студентов на систематическую подготовку к занятиям в течение семестра, что позволит использовать время экзаменационной сессии для систематизации знаний.

7. Оценочные средства

7.1 Паспорт оценочных средств экзамену

Семестр	Код контролируемой компетенции (или ее части)	Наименование оценочного средства
3	ПК-3	Тестовые задания по лекционному материалу. Вопросы по сдаче дисциплины. Отчеты по практическим занятиям.

7.2 Типовые задания или иные материалы, необходимые для текущего контроля

7.2.1 Типовые тестовые задания

Тема 1. Фундаментальные основы конструирования программного обеспечения

1. Детальное создание рабочей программной системы посредством комбинации кодирования, верификации, модульного тестирования, интеграционного тестирования и отладки называется

- конструирование ПО
- software construction
- ☐ программирование
- ☐ проектирование ПО
- ☐ моделирование ПО
- ☐ кодирование ПО

2. Процесс написания программного кода на выбранном языке программирования и соответствующей среде программирования называется

- кодирование ПО
- ☐ конструирование ПО
- ☐ software construction
- ☐ программирование
- ☐ проектирование ПО
- ☐ моделирование ПО

3. Подтверждение соответствия конечного продукта predetermined эталонным требованиям — это
- Верификация ПО
 - ☐ Валидация ПО
 - ☐ Тестирование ПО
 - ☐ Конструирование ПО
 - ☐ Проектирование ПО
4. Процесс подтверждения, что приведены доказательства того, что требования конкретного внешнего потребителя ПО удовлетворены называется
- ☐ Верификация ПО
 - Валидация ПО
 - ☐ Тестирование ПО
 - ☐ Конструирование ПО
 - ☐ Проектирование ПО
5. Процесс в программировании, позволяющий проверить на корректность отдельные блоки программы называется модульное тестирование
- unit testing
 - юнит-тестирование
 - ☐ отладка модуля
 - ☐ верификация модуля
 - ☐ валидация модуля|
6. Этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки называется
- ☐ unit testing
 - ☐ юнит-тестирование
 - debugging
 - отладка модуля
 - ☐ верификация модуля
 - ☐ валидация модуля|
7. Конструирование ПО связано наиболее полно с
- проектированием ПО
 - Software Design
 - тестированием ПО
 - Software Testing
 - ☐ программированием ПО
 - ☐ кодированием ПО
8. Фундаментальными основами конструирования ПО являются:
- Повторное использование кода
 - Стандарты в конструировании
 - ☐ Тестирование ПО
 - ☐ Кодирование ПО
 - ☐ Проектирование ПО

☐ Разработка требований

9. Фундаментальными основами конструирования ПО являются:

- Конструирование с возможностью проверки
- Повторное использование кода
- Стандарты в конструировании

- ☐ Тестирование ПО
- ☐ Кодирование ПО
- ☐ Проектирование ПО
- ☐ Разработка требований

10. Фундаментальными основами конструирования ПО являются:

- Стандарты в конструировании
- ☐ Тестирование ПО
- ☐ Кодирование ПО
- ☐ Проектирование ПО
- ☐ Разработка требований

11. Придание большей значимости читаемости кода и простоте тестирования — это

- минимизация сложности
- ☐ форматирование кода
- ☐ именование кода
- ☐ документирование кода

12. Уменьшение сложности в конструировании ПО достигается за счет

- создания простого кода
- легко читаемого кода
- ☐ производительного кода
- ☐ идеального кода
- ☐ именованного кода
- ☐ отлаженного кода

13. Построение ПО таким образом, чтобы сама система помогала вести поиск причин сбоев, будучи прозрачной для применения различных методов проверки называется

- конструирование с возможностью проверки
- ☐ программирование с возможностью проверки
- ☐ тестирование на лету
- ☐ модульное тестирование

14. Основная методология, которая применяется для сокращения трудозатрат при разработке сложных систем — это

- повторное использование
- ☐ использование библиотек программ
- ☐ применение фреймворков
- ☐ использование готовых модулей
- ☐ применение мастеров по созданию и настройке ПО

15. Направления хорошей реализации повторного использования:
- ☐ простое копирование части кода
 - оформление готовой ПО в виде подпрограммы или макроса с набором параметров
 - создание библиотек программ
 - ☐ сохранение «хороших» блоков программ
16. Повторимая архитектурная конструкция, представляющая собой решение проблемы разработки программы в рамках некоторой области применения называется
- шаблон проектирования
 - паттерн проектирования
 - ☐ фреймворк
 - ☐ библиотека программ
17. Детальное создание рабочей программной системы посредством комбинации кодирования, верификации, модульного тестирования, интеграционного тестирования и отладки называется
- конструирование ПО
 - software construction
 - ☐ программирование
 - ☐ проектирование ПО
 - ☐ моделирование ПО
 - ☐ кодирование ПО
18. Процесс написания программного кода на выбранном языке программирования и соответствующей среде программирования называется
- кодирование ПО
 - ☐ конструирование ПО
 - ☐ software construction
 - ☐ программирование
 - ☐ проектирование ПО
 - ☐ моделирование ПО
19. Подтверждение соответствия конечного продукта predetermined эталонным требованиям — это
- Верификация ПО
 - ☐ Валидация ПО
 - ☐ Тестирование ПО
 - ☐ Конструирование ПО
 - ☐ Проектирование ПО
20. Процесс подтверждения, что приведены доказательства того, что требования конкретного внешнего потребителя ПО удовлетворены называется
- ☐ Верификация ПО
 - Валидация ПО
 - ☐ Тестирование ПО
 - ☐ Конструирование ПО
 - ☐ Проектирование ПО

21. Процесс в программировании, позволяющий проверить на корректность отдельные блоки программы называется
модульное тестирование
- unit testing
 - юнит-тестирование
 - ☐ отладка модуля
 - ☐ верификация модуля
 - ☐ валидация модуля|
22. Этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки называется
- ☐ unit testing
 - ☐ юнит-тестирование
 - debugging
 - отладка модуля
 - ☐ верификация модуля
 - ☐ валидация модуля|
23. Конструирование ПО связано наиболее полно с
- проектированием ПО
 - Software Design
 - тестированием ПО
 - Software Testing
 - ☐ программированием ПО
 - ☐ кодированием ПО
24. Фундаментальными основами конструирования ПО являются:
- Повторное использование кода
 - Стандарты в конструировании
 - ☐ Тестирование ПО
 - ☐ Кодирование ПО
 - ☐ Проектирование ПО
 - ☐ Разработка требований
25. Фундаментальными основами конструирования ПО являются:
- Конструирование с возможностью проверки
 - Повторное использование кода
 - Стандарты в конструировании
 - ☐ Тестирование ПО
 - ☐ Кодирование ПО
 - ☐ Проектирование ПО
 - ☐ Разработка требований
26. Фундаментальными основами конструирования ПО являются:
- Стандарты в конструировании
 - ☐ Тестирование ПО
 - ☐ Кодирование ПО
 - ☐ Проектирование ПО

☐ Разработка требований

27. Придание большей значимости читаемости кода и простоте тестирования — это
- минимизация сложности
 - ☐ форматирование кода
 - ☐ именование кода
 - ☐ документирование кода

28. Уменьшение сложности в конструировании ПО достигается за счет
- создания простого кода
 - легко читаемого кода
 - ☐ производительного кода
 - ☐ идеального кода
 - ☐ именованного кода
 - ☐ отлаженного кода

29. Построение ПО таким образом, чтобы сама система помогала вести поиск причин сбоев, будучи прозрачной для применения различных методов проверки называется
- конструирование с возможностью проверки
 - ☐ программирование с возможностью проверки
 - ☐ тестирование на лету
 - ☐ модульное тестирование

30. Основная методология, которая применяется для сокращения трудозатрат при разработке сложных систем — это
- повторное использование
 - ☐ использование библиотек программ
 - ☐ применение фреймворков
 - ☐ использование готовых модулей
 - ☐ применение мастеров по созданию и настройке ПО

31. Стандарты в конструировании не включают:
- ☐ стандарты для форматов документа
 - ☐ языковые стандарты
 - ☐ стандарты кодирования
 - ☐ стандарты API
 - ☐ стандарты на инструменты конструирования
 - стандарты планирования
 - стандарты проектирования

32. Коммуникационные методы в конструировании — это стандарты
- для форматов документа
 - содержания документов
 - ☐ обмена данными по сети
 - ☐ упаковки данных в сетевые пакеты

33. соглашения по именованию идентификаторов относится к стандарту
- кодирования

- программирования
- ☐ конструирования
- ☐ проектирования
- ☐ разработки

34. Стандарты программных интерфейсов для вызовов функций операционной системы или среды называется

- платформой
- API
- SDK
- ☐ OMG
- ☐ UML
- ☐ MDA

35. Стандарты в конструировании ПО не создаются:

- ☐ консорциумами
- ☐ международными организациями по стандартизации
- ☐ производителями платформ
- ☐ производителями инструментов ПО
- производителями вычислительной техники
- производителями процессоров

36. Основным консорциумом в создании стандартов в конструировании ПО является

- OMG
- ☐ SWEEBOK
- ☐ W3C
- ☐ CORBA
- ☐ UML

37. К консорциуму OMG не относятся следующие наиболее употребительные стандарты

- ☐ CORBA
- ☐ UML
- ☐ MDA
- ☐ BPMN
- ☐ XMI
- XML
- W3C

38. UML (Unified Modeling Language) — унифицированный язык моделирования для графического описания объектного моделирования при разработке программного обеспечения. Позволяет ли UML детально проектировать пользовательский интерфейс системы?

- Нет
- ☐ Да
- ☐ Частично

39. На сегодняшнем этапе развития языков программирования наиболее используемыми являются

- PHP
- C++
- C#
- JavaScript

☐ Vbscript

☐ JScript

40. На сегодняшнем этапе развития языков программирования наиболее используемыми являются

☐ JScript

- Java

☐ Pascal

☐ Algol

☐ Fortran

☐ Perl

- PHP

41. Какое ПО предназначено для использования в ходе проектирования, разработки и сопровождения программ

- инструментальное

☐ прикладное

☐ системное

☐ резервное

42. Что из перечисленного осуществляет преобразование программы в форме исходного текста на языке ассемблера в машинные команды в виде объектного кода.

- ассемблеры

☐ трансляторы

☐ компоновщики

☐ отладчик

43. Что перечисленного выполняет преобразование программы с одного языка на другой.

- транслятор

☐ компоновщик

☐ компилятор

☐ ассемблер

44. Что из перечисленного переводит текст программы с языка высокого уровня, в эквивалентную программу на машинном языке.

- компилятор

☐ ассемблер

☐ интерпретатор

☐ компоновщик

45. Что из перечисленного анализирует команды или операторы программы и тут же выполняет их.
- интерпретатор
 - ☐ ассемблер
 - ☐ компоновщик
 - ☐ отладчик
46. Что из перечисленного принимает на вход один или несколько объектных модулей и собирает по ним исполняемый модуль.
- компоновщик
 - ☐ интерпретатор
 - ☐ ассемблер
 - ☐ компилятор
47. Что из перечисленного предназначено для поиска ошибок в программе.
- отладчик
 - ☐ ассемблер
 - ☐ компоновщик
 - ☐ интерпретатор
48. Что из перечисленного предназначено для создания и изменения текстовых файлов, а также их просмотра на экране, вывода на печать, поиска фрагментов текста и т. п.
- текстовый редактор
 - ☐ отладчик
 - ☐ компоновщик
 - ☐ интерпретатор
49. Что из перечисленного служит для создания и редактирования исходного кода программ.
- специализированные редакторы исходных текстов
 - ☐ программный редактор
 - ☐ редакторы графического интерфейса
 - ☐ графический редактор
50. Библиотеки подпрограмм — это
- сборники подпрограмм или объектов, используемых для разработки программного обеспечения
 - ☐ программы, используемые специально для ввода и редактирования текстовых данных
 - ☐ редакторы, предназначенные для работы с исходным кодом программы
 - ☐ программы для тестирования программного обеспечения
51. IDE - это
- система программных средств, используемых программистами для разработки ПО
 - ☐ сборники подпрограмм или объектов, используемых для разработки программного обеспечения
 - ☐ программы, используемые специально для ввода и редактирования текстовых данных

- ☐ комплект средств разработки, позволяющий создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы
52. Что из перечисленного является системой программных средств, используемых программистами для разработки ПО
- IDE
 - ☐ API
 - ☐ JDK
 - ☐ EJB
53. Что из перечисленного является комплектом средств разработки, позволяющим создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы.
- SDK
 - ☐ IDE
 - ☐ JDK
 - ☐ EJB
54. SDK - это
- ☐ система программных средств, используемых программистами для разработки ПО
 - ☐ сборники подпрограмм или объектов, используемых для разработки программного обеспечения
 - ☐ программы, используемые специально для ввода и редактирования текстовых данных
 - комплект средств разработки, позволяющий создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы
55. Парсеры и генераторы парсеров — это
- ПО для сопоставления линейной последовательности лексем с его формальной грамматикой. Результатом является дерево разбора (синтаксическое дерево). Обычно применяется совместно с лексическим анализом
 - ☐ пакет программ, позволяющих получать документацию по особым образом комментированному исходному коду и, в некоторых случаях, по исполняемым модулям.
 - ☐ программы для тестирования программного обеспечения
 - ☐ служат для организации автоматического тестирования разработанного ПО по заданным тестам
56. Генераторы документации - это
- ☐ ПО для сопоставления линейной последовательности лексем с его формальной грамматикой. Результатом является дерево разбора (синтаксическое дерево). Обычно применяется совместно с лексическим анализом
 - пакет программ, позволяющих получать документацию по особым образом комментированному исходному коду и, в некоторых случаях, по исполняемым модулям.
 - ☐ программы для тестирования программного обеспечения

- ☐ служат для организации автоматического тестирования разработанного ПО по заданным тестам
57. Средства анализа покрытия кода — это
- ☐ ПО для сопоставления линейной последовательности лексем с его формальной грамматикой. Результатом является дерево разбора (синтаксическое дерево). Обычно применяется совместно с лексическим анализом
 - ☐ пакет программ, позволяющих получать документацию по особым образом комментированному исходному коду и, в некоторых случаях, по исполняемым модулям.
 - программы для тестирования программного обеспечения
 - ☐ служат для организации автоматического тестирования разработанного ПО по заданным тестам
58. Программы для тестирования программного обеспечения - это
- Средства анализа покрытия кода
 - ☐ ПО для сопоставления линейной последовательности лексем с его формальной грамматикой. Результатом является дерево разбора (синтаксическое дерево). Обычно применяется совместно с лексическим анализом
 - ☐ пакет программ, позволяющих получать документацию по особым образом комментированному исходному коду и, в некоторых случаях, по исполняемым модулям
 - ☐ Парсеры и генераторы парсеров
59. Показать, насколько исходный код программы был протестирован это основная цель
- средств анализа покрытия кода
 - ☐ генераторов документаций
 - ☐ парсеров
 - ☐ генераторов парсеров
60. Что служит для организации автоматического тестирования разработанного ПО по заданным тестам
- средства автоматизированного тестирования
 - ☐ средства непрерывной интеграции
 - ☐ системы управления версиями
 - ☐ IDE
61. Что служит для выполнения частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.
- ☐ средства автоматизированного тестирования
 - средства непрерывной интеграции
 - ☐ системы управления версиями
 - ☐ IDE
62. Что служит для облегчения работы с изменяющейся информацией и позволяет хранить несколько версий одного и того же документа, при необходимости

возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

- ☐ средства автоматизированного тестирования
- ☐ средства непрерывной интеграции
 - системы управления версиями
- ☐ IDE

63. Что включает в себя интегрированная среда разработки(IDE, Integrated development environment)

- текстовый редактор
- компилятор и/или интерпретатор
- средства автоматизации сборки
- отладчик
- ☐ графический редактор

64. Что не включает в себя интегрированная среда разработки(IDE, Integrated development environment)

- ☐ текстовый редактор
- ☐ компилятор и/или интерпретатор
- ☐ средства автоматизации сборки
- ☐ отладчик
 - графический редактор

65. Что не включает в себя интегрированная среда разработки(IDE, Integrated development environment)

- ☐ текстовый редактор
- ☐ компилятор и/или интерпретатор
- ☐ средства автоматизации сборки
- ☐ отладчик
 - редактор конфигурационных файлов

66. Что не включает в себя интегрированная среда разработки(IDE, Integrated development environment)

- ☐ текстовый редактор
- ☐ компилятор и/или интерпретатор
- ☐ средства автоматизации сборки
- ☐ отладчик
 - веб-браузер

67. Что было создано для того, чтобы максимизировать производительность программиста благодаря тесно связанным компонентам с простыми пользовательскими интерфейсами

- IDE
- ☐ отладчик
- ☐ компилятор
- ☐ текстовый редактор

68. Отметьте функции текстовых редакторов:

- Написание исходного текста с учётом используемой кодировки
- Выполнение лексического анализа на лету
- ☐ строгое соблюдение правописания
- ☐ автоматическая обработка информации, представленной в текстовых файлах

69. Что из перечисленного не относится к функции текстовых редакторов:

- ☐ Написание исходного текста с учётом используемой кодировки
- ☐ Выполнение лексического анализа на лету
- строгое соблюдение правописания
- автоматическая обработка информации, представленной в текстовых файлах

70. Отметьте функции текстовых редакторов:

- Выполнение синтаксического анализа на лету
- Возможность реализации системы гиперссылок
- ☐ строгое соблюдение правописания
- ☐ автоматическая обработка информации, представленной в текстовых файлах

71. Отметьте функции компоновщика:

- связывание между собой объектных файлов и файлов библиотек
- пройти весь код, создаваемой программы, начиная с места вызова до её выхода.
- найти все вызовы внешних процедур и функций и увязать их с кодом других модулей, где описаны эти процедуры функции и переменные
- ☐ выполнение лексического анализа на лету
- ☐ выполнение синтаксического анализа на лету

72. Отметьте функции компоновщика:

- в современных системах компоновщик включает в выходной файл и ресурсы пользовательского интерфейса.
- если обнаруживается несоответствие между связями в выходной программе, то генерируется ошибка.
- ☐ выполнение лексического анализа на лету
- ☐ выполнение синтаксического анализа на лету

73. Что из перечисленного не относится к функциям компоновщика:

- ☐ в современных системах компоновщик включает в выходной файл и ресурсы пользовательского интерфейса.
- ☐ если обнаруживается несоответствие между связями в выходной программе, то генерируется ошибка.
- выполнение лексического анализа на лету
- выполнение синтаксического анализа на лету

74. В чем заключается функция загрузчика

- копирование исполняемого файла с диска в память и инициализация его выполнения.
- ☐ если обнаруживается несоответствие между связями в выходной программе, то генерируется ошибка.

- ☐ выполнение лексического анализа на лету
- ☐ выполнение синтаксического анализа на лету

75. В чем не заключается функция загрузчика

- ☐ копирование исполняемого файла с диска в память и инициализация его выполнения.
 - если обнаруживается несоответствие между связями в выходной программе, то генерируется ошибка.
 - выполнение лексического анализа на лету
 - выполнение синтаксического анализа на лету

76. Отметьте функции отладчика:

- последовательное пошаговое выполнение исходной программы на основе связанных с исходной программой созданных машинных команд.
- выполнение исходной программы до заданной точки прерывания.
- выполнение исходной программы до наступления заданных условий.
- ☐ если обнаруживается несоответствие между связями в выходной программе, то генерируется ошибка.

77. В чем не заключаются функции отладчика:

- ☐ последовательное пошаговое выполнение исходной программы на основе связанных с исходной программой созданных машинных команд.
- ☐ выполнение исходной программы до заданной точки прерывания.
- ☐ выполнение исходной программы до наступления заданных условий.
- если обнаруживается несоответствие между связями в выходной программе, то генерируется ошибка.

78. Отметьте функции отладчика:

- просмотр содержимого областей памяти используемой исходной программы при её выполнении.
- вычисление заданных выражений в исходной программе, по данным полученным в результате её выполнения.
- ☐ выполнение лексического анализа на лету
- ☐ выполнение синтаксического анализа на лету

79. Что включает в себя модель жизненного цикла

- разработка
- эксплуатация и сопровождение программного продукта
- ☐ время разработки
- ☐ время эксплуатации программного продукта

80. Что НЕ включает в себя модель жизненного цикла

- ☐ разработка
- ☐ эксплуатация и сопровождение программного продукта
- время разработки

- время эксплуатации программного продукта

81. Отметьте фазы жизненного цикла проекта

- инициализация(Initialization)
- планирование (Planning)
- ☐ реализация(realization)
- ☐ эксплуатация(exploitation)

82. Что из перечисленного не относится к фазам жизненного цикла проекта

- ☐ инициализация(Initialization)
- ☐ планирование (Planning)
- реализация(realization)
- эксплуатация(exploitation)

83. Сколько в основном фаз имеет жизненный цикл проекта

- 5
- ☐ 7
- ☐ 3
- ☐ 4

84. Отметьте фазы жизненного цикла проекта

- выполнение (Executing)
- завершение (Closing)
- контроль и мониторинг (Controlling and Monitoring)
- ☐ реализация(realization)
- ☐ эксплуатация(exploitation)

85. Что из перечисленного не относится к фазам жизненного цикла проекта

- ☐ выполнение (Executing)
- ☐ завершение (Closing)
- реализация(realization)
- эксплуатация(exploitation)

86. Жизненный цикл автоматизированной системы (АС) - это

- совокупность взаимосвязанных процессов создания и последовательного изменения состояния АС, от формирования исходных требований к ней до окончания эксплуатации и утилизации комплекса средств автоматизации АС
- ☐ последовательность фаз проекта, задаваемая исходя из потребностей управления проектом

87. Жизненный цикл проекта — это

- ☐ совокупность взаимосвязанных процессов создания и последовательного изменения состояния АС, от формирования исходных требований к ней до окончания эксплуатации и утилизации комплекса средств автоматизации АС
- последовательность фаз проекта, задаваемая исходя из потребностей управления проектом

88. Что определяет модель или парадигма жизненного цикла
- определяет концептуальный взгляд на организацию жизненного цикла и, часто, основные фазы жизненного цикла и принципы перехода между ними
 - ☐ совокупность взаимосвязанных процессов создания и последовательного изменения состояния АС, от формирования исходных требований к ней до окончания эксплуатации и утилизации комплекса средств автоматизации АС
 - ☐ последовательность фаз проекта, задаваемая исходя из потребностей управления проектом
89. Укажите модели жизненного цикла
- каскадная
 - спиральная
 - ☐ циклическая
 - ☐ повторяющаяся
90. Укажите модели жизненного цикла
- "водопадная"
 - спиральная
 - ☐ циклическая
 - ☐ повторяющаяся
91. Укажите модели жизненного цикла
- итеративная
 - спиральная
 - ☐ циклическая
 - ☐ повторяющаяся
92. Что из перечисленного не является моделями жизненного цикла
- ☐ итеративная
 - ☐ спиральная
 - циклическая
 - повторяющаяся
93. Каскадная (водопадная) или последовательная модель жизненного цикла - это
- работа над проектом движется линейно через ряд фаз
 - ☐ выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы
 - ☐ на каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество, и планируются работы следующего витка
 - ☐ разбиение большого объёма проектно-конструкторских работ на последовательность более малых составляющих частей
94. Итеративная и инкрементальная модель жизненного цикла – это
- ☐ работа над проектом движется линейно через ряд фаз

- выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы
 - ☐ на каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество, и планируются работы следующего витка
 - ☐ разбиение большого объёма проектно-конструкторских работ на последовательность более малых составляющих частей
95. Спиральная (spiral) модель жизненного цикла или модель Бозма
- ☐ работа над проектом движется линейно через ряд фаз
 - ☐ выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы
 - на каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество, и планируются работы следующего витка
 - ☐ разбиение большого объёма проектно-конструкторских работ на последовательность более малых составляющих частей
96. На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество, и планируются работы следующего витка - это характеристика
- ☐ Инкрементной модели жизненного цикла
 - Спиральной модели жизненного цикла
 - ☐ Последовательной модели жизненного цикла
 - ☐ Водопадной модели жизненного цикла
97. Инкрементная модель жизненного цикла
- ☐ работа над проектом движется линейно через ряд фаз
 - ☐ выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы
 - ☐ на каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество, и планируются работы следующего витка
 - разбиение большого объёма проектно-конструкторских работ на последовательность более малых составляющих частей
98. Разбиение большого объёма проектно-конструкторских работ на последовательность более малых составляющих частей характеризует
- Инкрементную модель жизненного цикла
 - ☐ Спиральную модель жизненного цикла
 - ☐ Последовательную модель жизненного цикла
 - ☐ Водопадную модель жизненного цикла

Тема 1.2 способностью составлять техническую документацию проектов автоматизации и информатизации прикладных процессов (ПК-9)

99. Количественные измерения, полученные в процессе и по результатам деятельности по конструированию ПО, называются
- Результаты аудита кода и метрики кода
 - ☐ Результаты профилирования
 - ☐ Отладочные результаты
 - ☐ Конечные данные
100. Мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций, это -
- Метрика программного обеспечения (software metric)
 - ☐ Количественная мера
 - ☐ Мера качества
 - ☐ Оценка соответствия стандартам
101. К классам метрик программного кода относятся
- Количественные метрики
 - Метрики сложности потока управления программы
 - ☐ Метрики Шеффера
 - ☐ Метрики корректности
102. К классам метрик программного кода относятся
- Количественные метрики
 - Метрики сложности потока управления программы
 - ☐ Метрики Шеффера
 - ☐ Метрики корректности
103. К классам метрик программного кода относятся
- Количественные метрики
 - Метрики сложности потока управления данными
 - ☐ Метрики Шеффера
 - ☐ Метрики корректности
104. К классам метрик программного кода относятся
- Метрики сложности потока управления программы
 - Метрики сложности потока управления данными
 - ☐ Метрики Шеффера
 - ☐ Метрики корректности
105. К классам метрик программного кода относятся
- Метрики оценок сложностей
 - Метрики сложности потока управления и данных программы
 - ☐ Метрики Шеффера
 - ☐ Метрики корректности
106. К количественным метрикам НЕ относится
- Метод Хансена

- ☐ Количество пустых строк
- ☐ Количество комментариев
- ☐ Процент комментариев

107. К количественным метрикам НЕ относится

- Мера Чена

- ☐ Количество пустых строк
- ☐ Количество комментариев
- ☐ Процент комментариев

108. К количественным метрикам НЕ относится

- Метрика Пивоварского

- ☐ Количество пустых строк
- ☐ Количество комментариев
- ☐ Процент комментариев

109. К количественным метрикам НЕ относится

- Мера Павлова

- ☐ Количество пустых строк
- ☐ Количество комментариев
- ☐ Процент комментариев

110. К количественным метрикам НЕ относится

- Цикломатическая сложность

- ☐ Количество пустых строк
- ☐ Количество комментариев
- ☐ Процент комментариев

111. К количественным метрикам НЕ относится

- Метрика Чепина

- ☐ Количество пустых строк
- ☐ Количество комментариев
- ☐ Процент комментариев

112. К количественным метрикам НЕ относится

- ☐ Среднее число строк для функций (классов, файлов).
- ☐ Среднее число строк, содержащих исходный код для функций (классов, файлов).
- ☐ Среднее число строк для модулей.
- Метод Хансена

113. К количественным метрикам НЕ относится

- ☐ Среднее число строк для функций (классов, файлов).
- ☐ Среднее число строк, содержащих исходный код для функций (классов, файлов).
- ☐ Среднее число строк для модулей.
- Метрика спена

114. К количественным метрикам НЕ относится

- ☐ Среднее число строк для функций (классов, файлов).
- ☐ Среднее число строк, содержащих исходный код для функций (классов, файлов).
- ☐ Среднее число строк для модулей.
 - Метрика обращений к глобальным переменным

115. К количественным метрикам НЕ относится

- ☐ Среднее число строк для функций (классов, файлов).
- ☐ Среднее число строк, содержащих исходный код для функций (классов, файлов).
- ☐ Среднее число строк для модулей.
 - Метрика Кафура

116. К количественным метрикам НЕ относится

- ☐ Среднее число строк для функций (классов, файлов).
- ☐ Среднее число строк, содержащих исходный код для функций (классов, файлов).
- ☐ Среднее число строк для модулей.
 - Цикломатическая сложность

117. К метрикам сложности потока управления программы НЕ относится

- Среднее число строк для модулей
- ☐ Метод Хансена
- ☐ Мера Чена
- ☐ Метрика Пивоварского

118. К метрикам сложности потока управления программы НЕ относится

- Среднее число строк для функций (классов, файлов)
- ☐ Метод Хансена
- ☐ Мера Чена
- ☐ Метрика Пивоварского

119. К метрикам сложности потока управления программы НЕ относится

- Среднее число строк, содержащих исходный код для функций (классов, файлов).
- ☐ Метод Хансена
- ☐ Мера Чена
- ☐ Метрика Пивоварского

120. К метрикам сложности потока управления программы НЕ относится

- Количество пустых строк
- ☐ Метод Хансена
- ☐ Мера Чена
- ☐ Метрика Пивоварского

121. К метрикам сложности потока управления программы НЕ относится

- Количество комментариев
- ☐ Метод Хансена
- ☐ Мера Чена
- ☐ Метрика Пивоварского

122. К метрикам сложности потока управления программы НЕ относится
- ☐ Процент комментариев
 - ☐ Метод Хансена
 - ☐ Мера Чена
 - ☐ Метрика Пивоварского
123. К метрикам сложности потока управления программы НЕ относится
- Среднее число строк для модулей
 - ☐ Метод Хансена
 - ☐ Мера Чена
 - ☐ Метрика Пивоварского
124. К метрикам сложности потока управления программы НЕ относится
- Среднее число строк для функций (классов, файлов)
 - ☐ Метод Хансена
 - ☐ Мера Чена
 - ☐ Метрика Пивоварского
125. К метрикам сложности потока управления программы НЕ относится
- Среднее число строк, содержащих исходный код для функций (классов, файлов).
 - ☐ Метрика Чепина
 - ☐ Метрика спена
 - ☐ Метрика обращений к глобальным переменным
126. К метрикам сложности потока управления программы НЕ относится
- Количество пустых строк
 - ☐ Метрика Чепина
 - ☐ Метрика спена
 - ☐ Метрика обращений к глобальным переменным
127. К метрикам сложности потока управления программы НЕ относится
- Количество комментариев
 - ☐ Метрика Чепина
 - ☐ Метрика спена
 - ☐ Метрика обращений к глобальным переменным
128. К метрикам сложности потока управления программы НЕ относится
- Процент комментариев
 - ☐ Метрика Чепина
 - ☐ Метрика спена
 - ☐ Метрика обращений к глобальным переменным
129. К метрикам сложности потока управления данными относится
- ☐ Процент комментариев
 - ☐ Количество комментариев

- ☐ Количество пустых строк
 - Метрика Кафура
- 130. ... — это число утверждений, содержащих данный идентификатор, между его первым и последним появлением в тексте программы
 - Спен
 - ☐ Цикл
 - ☐ Информационная сложность модуля
 - ☐ М-мера
- 131. Спен – это
 - число утверждений, содержащих данный идентификатор, между его первым и последним появлением в тексте программы
 - ☐ отношение фактического числа обращений к глобальным переменным к возможному
 - ☐ локализация обращений к данным внутри каждой программной секции
 - ☐ метрика на основе регулярных выражений
- 132. К метрикам сложности потока управления и данных программы относится
 - Тестирующая М-Мера
 - ☐ Метрика Чепина
 - ☐ Метрика спена
 - ☐ Метрика обращений к глобальным переменным
- 133. К метрикам сложности потока управления и данных программы относится
 - Метрика на основе регулярных выражений
 - ☐ Метрика Чепина
 - ☐ Метрика спена
 - ☐ Метрика обращений к глобальным переменным
- 134. К метрикам сложности потока управления и данных программы относится
 - Метрика Мак-Клура
 - ☐ Метрика Чепина
 - ☐ Метрика спена
 - ☐ Метрика обращений к глобальным переменным
- 135. Определение класса сложности задачи на основе анализа алгоритмов - это
 - Вычислительная сложность или порядок роста
 - ☐ Оценочная сложность
 - ☐ Мера связанности
 - ☐ Анализ функциональных точек
- 136. Стандартный метод измерения размера программного продукта, основанный на подсчете количества функционала, востребованного заказчиком и поставляемого разработчиком
 - Анализ функциональных точек
 - ☐ Плотность ошибочного кода

- ☐ М-мера
- ☐ Порядок роста

137. Отношение количества дефектов в коде к количеству строк кода

- Количество ошибок на 1000 строк кода

- ☐ М-мера
- ☐ Порядок роста
- ☐ Анализ функциональных точек

138. Отношение количества дефектных строк кода к их общему количеству

- Плотность ошибочного кода

- ☐ Вычислительная сложность
- ☐ Мера роста
- ☐ Степень покрытия

139. Показывает процент, насколько исходный код программы был протестирован

- Степень покрытия кода тестированием

- ☐ М-мера
- ☐ Степень покрытия
- ☐ Плотность ошибочного кода

140. К способам измерения степени покрытия кода тестированием относится

- Покрытие операторов

- ☐ М-мера
- ☐ Порядок роста
- ☐ Анализ функциональных точек

141. К способам измерения степени покрытия кода тестированием относится

- Покрытие условий

- ☐ М-мера
- ☐ Порядок роста
- ☐ Анализ функциональных точек

142. К способам измерения степени покрытия кода тестированием относится

- Покрытие путей

- ☐ М-мера
- ☐ Порядок роста
- ☐ Анализ функциональных точек

143. К способам измерения степени покрытия кода тестированием относится

- Покрытие функций

- ☐ М-мера
- ☐ Порядок роста
- ☐ Анализ функциональных точек

144. Выберите способ измерения степени покрытия коды, описывающийся следующим образом: “каждая ли строка исходного кода была выполнена и протестирована”
- Покрытие операторов
 - ☐ Покрытие условий
 - ☐ Покрытие путей
 - ☐ Покрытие функций
145. Выберите способ измерения степени покрытия коды, описывающийся следующим образом: “каждая ли точка решения (вычисления истинно ли или ложно выражение) была выполнена и протестирована”
- ☐ Покрытие операторов
 - Покрытие условий
 - ☐ Покрытие путей
 - ☐ Покрытие функций
146. Выберите способ измерения степени покрытия коды, описывающийся следующим образом: “все ли возможные пути через заданную часть кода были выполнены и протестированы”
- ☐ Покрытие операторов
 - ☐ Покрытие условий
 - Покрытие путей
 - ☐ Покрытие функций
147. Выберите способ измерения степени покрытия коды, описывающийся следующим образом: “каждая ли функция программы была выполнена”
- ☐ Покрытие операторов
 - ☐ Покрытие условий
 - ☐ Покрытие путей
 - Покрытие функций
148. К способам измерения степени покрытия кода тестированием относится
- Покрытие вход/выход
 - ☐ Оценочная сложность
 - ☐ Мера связанности
 - ☐ Анализ функциональных точек
149. К способам измерения степени покрытия кода тестированием относится
- Покрытие значений параметров
 - ☐ Оценочная сложность
 - ☐ Мера связанности
 - ☐ Анализ функциональных точек
150. К способам измерения степени покрытия кода тестированием относится
- Покрытие условий
 - ☐ Оценочная сложность
 - ☐ Мера связанности

- ☐ Анализ функциональных точек
151. К способам измерения степени покрытия кода тестированием относится
- Покрытие путей
 - ☐ Оценочная сложность
 - ☐ Мера связанности
 - ☐ Анализ функциональных точек
152. Выберите способ измерения степени покрытия кода, описывающийся следующим образом: “все ли вызовы функций и возвраты из них были выполнены”
- ☐ Покрытие операторов
 - ☐ Покрытие условий
 - ☐ Покрытие путей
 - Покрытие вход/выход
153. Выберите способ измерения степени покрытия кода, описывающийся следующим образом: “все ли типовые и граничные значения параметров были проверены”
- ☐ Покрытие операторов
 - ☐ Покрытие условий
 - Покрытие значений параметров
 - ☐ Покрытие вход/выход
154. ... позволяет оценить степень полноты системы тестов по отношению к функциональности системы
- Покрытие требований
 - ☐ Количество классов и интерфейсов
 - ☐ Анализ функциональных точек
 - ☐ М-мера
155. Рефакторинг (Refactoring) - это
- Изменение во внутренней структуре программного обеспечения, имеющее целью облегчить понимание его работы и упростить модификацию, не затрагивая наблюдаемого поведения
 - ☐ Изменение функциональности программы с целью соответствия выдвигаемым требованиям
 - ☐ Выбор алгоритмов и методик тестирования программного кода
 - ☐ Проектирование нового программного кода на основе уже существующего
156. Изменение во внутренней структуре программного обеспечения, имеющее целью облегчить понимание его работы и упростить модификацию, не затрагивая наблюдаемого поведения
- Рефакторинг
 - ☐ Анализ требований
 - ☐ Реинжиниринг
 - ☐ Дизайн
157. Цель рефакторинга —
- Сделать код программы легче для понимания

- ☐ Увеличить быстродействие кода
- ☐ Сконструировать тестовые модули
- ☐ Изменить код для увеличения числа строк

158. К видимым проблемам в коде, требующим рефакторинга, НЕ относится

- наличие комментариев;
- ☐ длинный метод;
- ☐ большой класс;
- ☐ длинный список параметров;

159. К видимым проблемам в коде, требующим рефакторинга, НЕ относится

- наличие локальных переменных;
- ☐ длинный метод;
- ☐ большой класс;
- ☐ длинный список параметров;

160. К видимым проблемам в коде, требующим рефакторинга, НЕ относится

- наличие модульности;
- ☐ длинный метод;
- ☐ большой класс;
- ☐ длинный список параметров;

161. К видимым проблемам в коде, требующим рефакторинга, НЕ относится

- организованная правильным образом иерархия классов;
- ☐ длинный метод;
- ☐ большой класс;
- ☐ длинный список параметров;

162. К видимым проблемам в коде, требующим рефакторинга, НЕ относится

- разветвление программы
- ☐ противоположность расходящейся модификации (внесение множества мелких изменений в большое число классов)
- ☐ «завистливые» функции (метод, который чрезмерно обращается к данным другого объекта)
- ☐ группы данных - группировку данных вынести в отдельный класс;

163. К видимым проблемам в коде, требующим рефакторинга, НЕ относится

- операторы if;
- ☐ одержимость элементарными типами;
- ☐ операторы switch;
- ☐ параллельные иерархии наследования (при порождении подкласса одного из классов приходится создавать подкласс другого класса);

164. К видимым проблемам в коде, требующим рефакторинга, НЕ относится

- ☐ ленивый класс - класс, существование которого не окупается выполняемыми им функциями

- ☐ теоретическая общность — код существует, как задел на будущее
- ☐ временное поле - в некотором объекте атрибут устанавливается только при определенных обстоятельствах
- напряженное поле

165. Цепочки сообщений -

- делегация параметров через несколько классов
- ☐ класс делегирует обработку другому классу
- ☐ слишком тесная связь двух классов
- ☐ дублирование функций в двух классах

166. Посредник

- класс делегирует обработку другому классу
- ☐ слишком тесная связь двух классов
- ☐ дублирование функций в двух классах
- ☐ делегация параметров через несколько классов

167. Неуместная близость

- слишком тесная связь двух классов
- ☐ дублирование функций в двух классах
- ☐ делегация параметров через несколько классов
- ☐ класс делегирует обработку другому классу

168. Альтернативные классы с разными интерфейсами

- дублирование функций в двух классах
- ☐ делегация параметров через несколько классов
- ☐ класс делегирует обработку другому классу
- ☐ слишком тесная связь двух классов

169. Классы данных содержат:

- только поля и способы доступа к этим полям
- ☐ поля
- ☐ способы доступа к полям
- ☐ ничего из перечисленного не содержат

170. О чем НЕ свидетельствуют комментарии

- ☐ о непонятности кода
- о простоте кода
- ☐ о незавершенности кода
- ☐ о недоработках в коде

171. Выберите верное описание термина “изменение сигнатуры метода” (Change Method Signature)

- заключается в добавлении, изменении или удалении параметра метода
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами

- ☐ несколько клиентов пользуются одним и тем же подмножеством интерфейса класса или в двух классах часть интерфейса является общей
- ☐ на основе выражения создает параметр метода

172. Выберите верное описание термина “Инкапсуляция поля” (Encapsulate Field)

- если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
- ☐ несколько клиентов пользуются одним и тем же подмножеством интерфейса класса или в двух классах часть интерфейса является общей
- ☐ на основе выражения создает параметр метода

173. Выберите верное описание термина “Выделение класса” (Extract Class)

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- некоторый класс выполняет работу, которую следует поделить между двумя классами
- ☐ несколько клиентов пользуются одним и тем же подмножеством интерфейса класса или в двух классах часть интерфейса является общей
- ☐ на основе выражения создает параметр метода

174. Выберите верно описание термина “Выделение интерфейса” (Extract Interface)

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
- несколько клиентов пользуются одним и тем же подмножеством интерфейса класса или в двух классах часть интерфейса является общей. Необходимо выделить это подмножество в интерфейс
- ☐ на основе выражения создает параметр метода

?Выберите верно описание термина “ Выделение локальной переменной (Extract Local Variable)”

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
- берет выражение, которое используется непосредственно, и сначала присваивает его значение локальной переменной. Эта переменная затем используется там, где использовалось выражение.
- ☐ на основе выражения создает параметр метода

175. Выберите верно описание термина “Выделение метода (Extract Method)”

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
- заключается в выделении из длинного и/или требующего комментариев кода отдельных фрагментов и преобразовании их в отдельные методы, с подстановкой подходящих вызовов в местах использования
- ☐ на основе выражения создает параметр метода

176. Если фрагмент кода требует комментария о том, что он делает, то он
- должен быть выделен в отдельный метод
 - ☐ должен быть перемещен в другой метод
 - ☐ должен быть удален
 - ☐ должен быть выделен в класс
177. Один метод по правилам рефакторинга не должен занимать более чем
- один экран (25-50 строк)
 - ☐ один байт
 - ☐ один такт процессора
 - ☐ один мегабайт
178. Выберите верно описание термина “Генерализация типа (Generalize Type)”
- идея в том, чтобы использовать преимущества объектно-ориентированного программирования и сделать более обобщенные типы, позволяя выполнить совместное использование кода, что приводит к упрощению программы
 - ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
 - ☐ несколько клиентов пользуются одним и тем же подмножеством интерфейса класса или в двух классах часть интерфейса является общей. Необходимо выделить это подмножество в интерфейс
 - ☐ на основе выражения создает параметр метода.
179. Выберите верно описание термина “Встраивание (Inline)”
- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
 - ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
 - задание функции, как встроенной с помощью модификатора inline, который рекомендует компилятору вместо обращения к функции помещать ее код непосредственно в каждую точку вызова во время компиляции
 - ☐ на основе выражения создает параметр метода
180. Модификатор inline
- рекомендует компилятору вместо обращения к функции помещать ее код непосредственно в каждую точку вызова во время компиляции
 - ☐ позволяет вынести функцию из класса
 - ☐ позволяет внести функцию в класс
 - ☐ модифицирует уровень доступа к методу
181. Модификатор inline ставится
- перед типом функции
 - ☐ перед именем функции
 - ☐ перед уровнем доступа функции
 - ☐ после имени функции
182. Модификатор inline применяется для
- коротких функций, чтобы снизить накладные расходы на вызов.

- ☐ длинный функций, чтобы ускорить их выполнение
- ☐ коротких функций, чтобы выделить их из класса
- ☐ любых функций, чтобы выделить их из класса

183. Выберите верно описание термина "Введение параметра (Introduce Parameter)"

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
- ☐ задание функции, как встроенной с помощью модификатора inline, который рекомендует компилятору вместо обращения к функции помещать ее код непосредственно в каждую точку вызова во время компиляции
- на основе выражения создает параметр метода

184. Выберите верно описание термина "Спуск метода (Push Down Method)"

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
- перемещает выделенные методы и поля из класса в его подклассы.
- ☐ на основе выражения создает параметр метода

185. Выберите верно описание термина "Подъем метода (Pull Up Method)"

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
- выполняет операцию, обратную операции Push Down.
- ☐ на основе выражения создает параметр метода

186. Выберите верно описание термина "Переименование (Rename)"

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
- переименовывает выбранный элемент с корректировкой ссылок с использованием быстрых клавиш
- ☐ на основе выражения создает параметр метода

187. Выберите верно описание термина "Перемещение метода (Move Method)"

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
- ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
- ☐ несколько клиентов пользуются одним и тем же подмножеством интерфейса класса или в двух классах часть интерфейса является общей. Необходимо выделить это подмножество в интерфейс
- перемещает выбранный элемент с корректировкой ссылок с использованием быстрых клавиш.

188. Выберите верно описание термина "Замена условного оператора полиморфизмом (Replace Conditional with Polymorphism)"

- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
 - ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
 - есть условный оператор, поведение которого зависит от типа объекта. Переместите каждую ветвь условного оператора в перегруженный метод подкласса. Сделайте исходный метод абстрактным.
 - ☐ на основе выражения создает параметр метода
189. Выберите верно описание термина “Замена наследования делегированием (Replace Inheritance with Delegation)”
- ☐ если у класса имеется открытое поле, необходимо сделать его закрытым и обеспечить методы доступа
 - ☐ некоторый класс выполняет работу, которую следует поделить между двумя классами
 - подкласс использует только часть интерфейса родительского класса или не желает наследовать данные. Создайте поле для родительского класса, настройте методы, чтобы они делегировали выполнение родительскому классу, и удалите подклассы.
 - ☐ на основе выражения создает параметр метода
190. Все современные интегрированные среды программирования (IDE) имеют
- базовый набор методов рефакторинга.
 - ☐ графический редактор
 - ☐ аудио анализатор
 - ☐ генератор формул
191. Атомарная отмена операции незаменима в том случае, если
- изменения вносятся сразу в нескольких участках кода.
 - ☐ программа работает некорректно
 - ☐ в код не вносятся никаких изменений
 - ☐ в ходе выполнения программы возникла ошибка
192. Практически во всех интегрированных средах реализован рефакторинг для языка
- Java
 - ☐ Pascal
 - ☐ Ruby
 - ☐ Pearl
193. Конфигурация программного обеспечения — это
- совокупность настроек программы, задаваемая пользователем.
 - ☐ описание работы кода
 - ☐ набор символов или последовательностей, вставляемых в текст для передачи информации о его выводе
 - ☐ текстовый документ, написанный с использованием языка разметки
194. Многие программы хранят настройки в
- текстовых файлах, имеющие свои форматы, зависящие от языка конфигурирования.
 - ☐ аудиофайлах
 - ☐ исполняемом файле
 - ☐ файле регистрации

195. Набор символов или последовательностей, вставляемых в текст для передачи информации о его выводе или строении – это
- язык разметки
 - ☐ сценарий
 - ☐ XML-процессор
 - ☐ система компьютерной алгебры
196. Текстовый документ, написанный с использованием языка разметки, содержит
- не только сам текст, но и дополнительную информацию о различных его участках.
 - ☐ только текст
 - ☐ ссылки
 - ☐ расшифровку сокращений, используемых в тексте
197. Построчное написание параметров с заданием своих ключевых лексем осуществляется в файле с расширением
- ini
 - ☐ oni
 - ☐ ino
 - ☐ nin
198. Текстовый формат обмена данными, основанный на JavaScript
- JSON
 - ☐ ini
 - ☐ XML
 - ☐ YAML
199. Человеческочитаемый формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования
- YAML
 - ☐ JSON
 - ☐ XML
 - ☐ ini
200. Расширяемый язык разметки, рекомендованный Консорциумом Всемирной паутины (W3C)
- XML
 - ☐ JSON
 - ☐ YAML
 - ☐ ini
201. Программы, читающие XML-документы и обеспечивающие доступ к их содержимому
- XML-процессоры
 - ☐ графические редакторы
 - ☐ математические системы
 - ☐ сценарии
202. Конфигурация чаще всего имеет
- древовидную структуру
 - ☐ циклическую структуру
 - ☐ линейную структуру
 - ☐ математическую структуру

203. Всё, что следует за этим символом в конфигурационном файле, является комментарием и игнорируется
- #
 - ☐ *
 - ☐ &
 - ☐ !
204. Синтаксический анализатор в виде программы, выполняющий синтаксический анализ, используемый для чтения конфигурационных файлов
- парсер
 - ☐ инструментальный язык
 - ☐ сценарный язык
 - ☐ сценарий
205. Язык конструирования из повторно-используемых элементов
- ☐ парсер
 - инструментальный язык
 - ☐ сценарный язык
 - ☐ сценарий
206. Высокоуровневый язык программирования для написания сценариев — кратких описаний действий, выполняемых системой
- ☐ парсер
 - ☐ инструментальный язык
 - сценарный язык
 - ☐ сценарий
207. Программа, имеющая дело с готовыми программными компонентами
- ☐ парсер
 - ☐ инструментальный язык
 - ☐ сценарный язык
 - сценарий
208. Служат для управления заданиями в операционных системах. Эти языки чаще всего используются в пакетном режиме обработки.
- командно-сценарные языки
 - ☐ прикладные сценарные языки
 - ☐ языки разметки
 - ☐ универсальные сценарные языки
209. Реализуют интерактивное общение с ОС. В клиент-серверной архитектуре такие языки работают в клиентской части программного обеспечения.
- ☐ командно-сценарные языки
 - прикладные сценарные языки
 - ☐ языки разметки
 - ☐ универсальные сценарные языки
210. Языки для встраивания специальных кодов (тегов) в обычный текст не только для целей структурирования и форматирования, но и для определения динамического поведения.
- ☐ командно-сценарные языки
 - ☐ прикладные сценарные языки
 - языки разметки
 - ☐ универсальные сценарные языки

211. Эти языки чаще всего применяются для программирования для веб страниц.

- ☐ командно-сценарные языки
- ☐ прикладные сценарные языки
- ☐ языки разметки
 - универсальные сценарные языки

212. Командно-сценарным языком является

- bash
- ☐ AutoLISP
- ☐ VBA
- ☐ XML

213. Командно-сценарным языком является

- COMMAND
- ☐ AutoLISP
- ☐ VBA
- ☐ XML

214. Командно-сценарным языком является

- PowerShell
- ☐ AutoLISP
- ☐ VBA
- ☐ XML

215. Командно-сценарным языком является

- VB Script
- ☐ AutoLISP
- ☐ VBA
- ☐ XML

216. Прикладным сценарным языком является

- ☐ VB Script
- AutoLISP
- ☐ PHP
- ☐ XML

217. Прикладным сценарным языком является

- ☐ VB Script
- Emacs Lisp
- ☐ PHP
- ☐ XML

218. Прикладным сценарным языком является

- ☐ VB Script
- ERM
- ☐ PHP
- ☐ XML

219. Прикладным сценарным языком является

- ☐ VB Script
- UnrealScript
- ☐ PHP
- ☐ XML

220. Прикладным сценарным языком является

- ☐ VB Script
 - JavaScript
 - ☐ PHP
 - ☐ XML
221. Языком разметки является
- ☐ VB Script
 - ☐ JavaScript
 - ☐ PHP
 - XML
222. Языком разметки является
- ☐ VB Script
 - ☐ JavaScript
 - ☐ PHP
 - GML
223. Языком разметки является
- ☐ VB Script
 - ☐ JavaScript
 - ☐ PHP
 - TeX
224. Языком разметки является
- ☐ VB Script
 - ☐ JavaScript
 - ☐ PHP
 - SGML
225. Языком разметки является
- ☐ VB Script
 - ☐ JavaScript
 - ☐ PHP
 - PostScript
226. Языком разметки является
- ☐ VB Script
 - ☐ JavaScript
 - ☐ PHP
 - MathML
227. Универсальным сценарным языком является
- ☐ VB Script
 - ☐ JavaScript
 - Tcl
 - ☐ MathML
228. Универсальным сценарным языком является
- ☐ VB Script
 - ☐ JavaScript
 - Lua
 - ☐ MathML
229. Универсальным сценарным языком является
- ☐ VB Script

- ☐ JavaScript
 - Perl
 - ☐ MathML
230. Универсальным сценарным языком является
- ☐ VB Script
 - ☐ JavaScript
 - Python
 - ☐ MathML
231. Универсальным сценарным языком является
- ☐ VB Script
 - ☐ JavaScript
 - REBOL
 - ☐ MathML
232. Универсальным сценарным языком является
- ☐ VB Script
 - ☐ JavaScript
 - Ruby
 - ☐ MathML
233. В отличие от плагинов, скрипты
- интерпретируются
 - ☐ компилируются
 - ☐ не делают ничего
 - ☐ дифференцируются
234. Неправильно написанный скрипт
- выведет диагностическое сообщение, а не приведёт к системному краху.
 - ☐ не выведет ничего
 - ☐ приведет к системному краху
 - ☐ выведет "Hello world!"
235. На скриптовом языке может писать
- программист очень низкой квалификации
 - ☐ программист только самой высокой квалификации
 - ☐ только машина
 - ☐ никто, скрипт всегда пишется автоматически
236. Формальная знаковая система, предназначенная для записи компьютерных программ
- язык программирования
 - ☐ язык разметки
 - ☐ произвольный набор символов
 - ☐ язык математический
237. Этот язык определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под её управлением.
- язык программирования
 - ☐ язык разметки
 - ☐ произвольный набор символов
 - ☐ язык математический

238. Этот язык предназначен для написания компьютерных программ, которые применяются для передачи компьютеру инструкций (команд) по выполнению того или иного вычислительного процесса и организации управления отдельными устройствами.
- язык программирования
 - ☐ язык разметки
 - ☐ произвольный набор символов
 - ☐ язык математический
239. Множество символов, используемых для представления понятий и их взаимоотношений, используемых при определении языков программирования.
- нотация
 - ☐ язык разметки
 - ☐ произвольный набор символов
 - ☐ система компьютерной алгебры
240. Сколько существует основных видов нотаций
- 3
 - ☐ 6
 - ☐ 1
 - ☐ 5
241. Какого основного вида нотаций не существует для языков программирования
- акустическая
 - ☐ лингвистическая
 - ☐ формальная
 - ☐ визуальная
242. Какого основного вида нотаций не существует для языков программирования
- диетическая
 - ☐ лингвистическая
 - ☐ формальная
 - ☐ визуальная
243. Какого основного вида нотаций не существует для языков программирования
- непрерывная
 - ☐ лингвистическая
 - ☐ формальная
 - ☐ визуальная
244. Эта нотация характеризуется использованием строк текста, содержащих специализированные “слова”, представляющие сложные программные конструкции и комбинируемые в шаблоны, напоминающие предложения, построенные в соответствии с определенным синтаксисом
- ☐ непрерывная
 - лингвистическая
 - ☐ формальная
 - ☐ визуальная
245. строгая смысловая нагрузка, обеспечивающая интуитивное понимание того, что будет происходить когда будет выполняться программное обеспечение
- семантика
 - ☐ математика
 - ☐ лингвистика
 - ☐ лексика

246. Программное приложение для символьных вычислений, то есть выполнения преобразований и работы с математическими выражениями в аналитической (символьной) форме
- система компьютерной алгебры
 - ☐ лексический анализатор
 - ☐ генератор формул
 - ☐ логический контроллер
247. К системам компьютерной алгебры относят
- Matchcad
 - ☐ JavaScript
 - ☐ Paint
 - ☐ Linux Mint
248. К системам компьютерной алгебры не относится
- Cisco Packet Tracer
 - ☐ Mathcad
 - ☐ Mathematica
 - ☐ MATLAB
249. К основным этапам кодирования не относится
- Внедрение приложения
 - ☐ Предварительное решение задачи методами математики
 - ☐ Определение методов решения полученных математических выражений на основе ЭВМ с помощью дискретной математики
 - ☐ Разработка алгоритма решения задачи
250. К основным этапам кодирования не относится
- Анализ рынка
 - ☐ Предварительное решение задачи методами математики
 - ☐ Определение методов решения полученных математических выражений на основе ЭВМ с помощью дискретной математики
 - ☐ Разработка алгоритма решения задачи
251. Набор правил и соглашений, используемых при написании исходного кода на некотором языке программирования. Обычно принимается и используется некоторой группой разработчиков программного обеспечения для единообразного оформления совместно используемого кода.
- стандарт оформления кода
 - ☐ семантика
 - ☐ парсер
 - ☐ сценарий
252. Если программа плохо спроектирована, слабо структурирована, запутанная и трудная для понимания, содержит операторы GOTO, исключений и других конструкций, ухудшающих структурированность — то она называется
- спагетти-код
 - ☐ хэш код
 - ☐ верблюжий код
 - ☐ код Олмана
253. Регистр символов, когда все символы маленькие
- нижний
 - ☐ верхний

- ☐ «верблюжий»
 - ☐ «верблюжий» с малой буквы
254. Регистр символов, когда все символы прописные
- ☐ нижний
 - верхний
 - ☐ «верблюжий»
 - ☐ «верблюжий» с малой буквы
255. Регистр символов, когда несколько слов пишутся слитно без пробелов, при этом каждое слово пишется с заглавной буквы
- ☐ нижний
 - ☐ верхний
 - «верблюжий»
 - ☐ «верблюжий» с малой буквы
256. Регистр символов, когда первое слово пишется с малой буквы
- ☐ нижний
 - ☐ верхний
 - ☐ «верблюжий»
 - «верблюжий» с малой буквы
257. Для улучшения читаемости исходного текста программы рекомендуется писать
- не более одного оператора в строке
 - ☐ не менее четырёх операторов в строке
 - ☐ не менее трёх операторов в строке
 - ☐ не более пяти операторов в строке
258. Использование пробелов при оформлении логических и арифметических выражений не корректно в следующем случае
- пробел не должен стоять после запятой в списке аргументов
 - ☐ ключевое слово и следующая за ним открывающая скобка должны быть разделены пробелом
 - ☐ пробелы не должны разделять название метода и следующую за ним открывающую скобку
 - ☐ все бинарные операторы исключая . должны быть разделены при помощи пробела
259. Использование пробелов при оформлении логических и арифметических выражений не корректно в следующем случае
- ☐ пробел должен стоять после запятой в списке аргументов
 - ☐ ключевое слово и следующая за ним открывающая скобка должны быть разделены пробелом
 - ☐ пробелы не должны разделять название метода и следующую за ним открывающую скобку
 - все бинарные операторы исключая . не должны быть разделены при помощи пробела
260. Использование пробелов при оформлении логических и арифметических выражений не корректно в следующем случае
- ☐ пробел должен стоять после запятой в списке аргументов
 - ключевое слово и следующая за ним открывающая скобка не должны быть разделены пробелом
 - ☐ пробелы не должны разделять название метода и следующую за ним открывающую скобку
 - ☐ все бинарные операторы исключая . должны быть разделены при помощи пробела

261. Использование пробелов при оформлении логических и арифметических выражений не корректно в следующем случае
- ☐ пробел должен стоять после запятой в списке аргументов
 - ☐ ключевое слово и следующая за ним открывающая скобка должны быть разделены пробелом
 - пробелы должны разделять название метода и следующую за ним открывающую скобку
 - ☐ все бинарные операторы исключая . должны быть разделены при помощи пробела
262. Использование пустых строк является важным средством для выделения участков программы. При этом не имеет смысла отделять
- однотипные инструкции или директивы
 - ☐ определения переменных от других частей кода
 - ☐ последовательности однотипных инструкций или директив
 - ☐ функции
263. Участок кода, который программист оставляет незавершённым, чтобы вернуться к нему позже обычно помечают комментарием
- TODO
 - ☐ FIXME (KLUDGE)
 - ☐ XXX или ZZZ
 - ☐ DON'T DO
264. Этим комментарием обычно помечают обнаруженную ошибку, которую решают исправить позже
- ☐ TODO
 - FIXME (KLUDGE)
 - ☐ XXX или ZZZ
 - ☐ DON'T DO
265. Этим комментарием обычно обозначают найденную критическую ошибку, без исправления которой нельзя продолжать дальнейшую работу
- ☐ TODO
 - ☐ FIXME (KLUDGE)
 - XXX или ZZZ
 - ☐ DON'T DO
266. Что такое Интерфейс прикладного программирования (API)
- набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.
 - ☐ общая граница между двумя функциональными объектами, требования к которой определяются стандартом; совокупность средств, методов и правил взаимодействия (управления, контроля и т.д.) между элементами системы
 - ☐ разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений
 - ☐ разновидность интерфейса пользователя, использующая при вводе-выводе и представлении информации исключительно набор буквенно-цифровых символов и символов псевдографики
267. Что определяет API

- функциональность, которую предоставляет программа.
 - ☐ быстроедействие, которое предоставляет программа
 - ☐ графический интерфейс программы
 - ☐ отказоустойчивость программы
268. Что включает в себя API библиотеки функций и классов
- описание сигнатур и семантики функций.
 - ☐ описание пользовательского интерфейса
 - ☐ описание математических функций
 - ☐ описание семантики функций
269. Что такое сигнатура функции –
- часть общего объявления функции, позволяющая средствам трансляции идентифицировать функцию среди других.
 - ☐ часть общего объявления класса, позволяющая средствам трансляции идентифицировать функцию среди других
 - ☐ часть частного объявления класса, позволяющая средствам трансляции идентифицировать функцию среди других
 - ☐ часть частного объявления функции, позволяющая средствам трансляции идентифицировать функцию среди других
270. Часть общего объявления функции, позволяющая средствам трансляции идентифицировать функцию среди других
- сигнатура функции.
 - ☐ семантика функции
 - ☐ математических функций
 - ☐ пользовательского интерфейса
271. Что такое семантика функции –
- описывает то, что делает данная функция
 - ☐ описывает то, что делает данный класс
 - ☐ описывает то, что делает данная программа
 - ☐ описывает то, что делает данный объект класса
272. Главный API операционных систем – это
- множество системных вызовов.
 - ☐ обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции
 - ☐ множество сетевых вызовов
 - ☐ множество локальных вызовов
273. Системный вызов (system call) –
- обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.
 - ☐ обращение прикладной программы к классу программы для выполнения какой-либо операции
 - ☐ обращение прикладной программы к функции программы для выполнения какой-либо операции
 - ☐ обращение ядра операционной системы к объекту программы для выполнения какой-либо операции
274. WxWidgets - это
- кроссплатформенная библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений.

- ☐ кроссплатформенный инструментарий разработки ПО на языке программирования C++
 - ☐ кроссплатформенная библиотека элементов интерфейса, имеет простой в использовании API
 - ☐ библиотека для ввода и вывода на языке программирования C++
275. Qt - это
- кроссплатформенный инструментарий разработки ПО на языке программирования C++.
 - ☐ кроссплатформенная библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений
 - ☐ кроссплатформенная библиотека элементов интерфейса, имеет простой в использовании API
 - ☐ библиотека для ввода и вывода на языке программирования C++
276. GTK - это
- кроссплатформенная библиотека элементов интерфейса, имеет простой в использовании API.
 - ☐ кроссплатформенный инструментарий разработки ПО на языке программирования C++
 - ☐ кроссплатформенная библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений
 - ☐ библиотека для ввода и вывода на языке программирования C++
277. Написание библиотек, которые отображают системные вызовы одной ОС в системные вызовы другой ОС
- Wine
 - ☐ Qt
 - ☐ GTK
 - ☐ WxWidgets
278. Написание библиотек, которые отображают системные вызовы одной ОС в системные вызовы другой ОС
- Cygwin
 - ☐ Qt
 - ☐ GTK
 - ☐ WxWidgets
279. Классификация API графических интерфейсов
- Direct3D
 - DirectDraw
 - ☐ DirectSound
 - ☐ API Википедия
280. Классификация API графических интерфейсов
- OpenVG
 - OpenGL
 - ☐ DirectSound
 - ☐ API Википедия
281. Классификация API звуковых интерфейсов
- ☐ OpenVG
 - ☐ OpenGL
 - DirectSound
 - ☐ API Википедия
282. Классификация API звуковых интерфейсов

- ☐ OpenVG
 - DirectMusic
 - DirectSound
- ☐ API Википедия
- 283. Для разработки приложений с использованием собственных библиотек
 - Delphi
 - Visual Studio
- ☐ Android
- ☐ PHP
- 284. Для мобильных систем
 - ☐ Delphi
 - ☐ Visual Studio
 - Android
 - ☐ PHP
- 285. Для WEB приложений
 - ☐ Delphi
 - ☐ Visual Studio
 - ☐ Android
 - PHP
- 286. Что за программа Wine
 - свободное программное обеспечение, позволяющее пользователям UNIX-подобных систем архитектуры x86 исполнять 16-, 32- и 64- битные приложения Microsoft Windows.
 - ☐ UNIX-подобная среда и интерфейс командной строки для Microsoft Windows
 - ☐ кроссплатформенный инструментарий разработки ПО на языке программирования C++
 - ☐ кроссплатформенный инструментарий разработки интерфейса на языке программирования C++
- 287. Что за программа Cygwin
 - UNIX-подобная среда и интерфейс командной строки для Microsoft Windows.
 - ☐ свободное программное обеспечение, позволяющее пользователям UNIX-подобных систем архитектуры x86 исполнять 16-, 32- и 64- битные приложения Microsoft Windows
 - ☐ кроссплатформенный инструментарий разработки ПО на языке программирования C++
 - ☐ кроссплатформенный инструментарий разработки интерфейса на языке программирования C++
- 288. GNU C Library (Glibc) – это
 - является библиотекой Си, которая обеспечивает системные вызовы и основные функции, используемые в Linux.
 - ☐ кроссплатформенная библиотека элементов интерфейса, имеет простой в использовании API
 - ☐ кросс-платформенная библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений, в частности для построения графического интерфейса пользователя (GUI)
 - ☐ означает коллекцию классов и функций, для работы с этими контейнерами, объектов-функций, основных типов строк и потоков
- 289. Платформа – это

- совокупность взаимодействующих между собой аппаратных средств и операционной системы, под управлением которой функционируют прикладные программы и средства для их разработки.
- ☐ совокупность взаимодействующих между многопоточных программ и операционной системы
- ☐ совокупность взаимодействующих между собой кроссплатформенных библиотек и операционной системы, под управлением которой функционируют прикладные программы и средства для их разработки
- ☐ совокупность операционной системы, средств разработки прикладных программных решений и прикладных программ, работающих под управлением данной операционной системы

290. Что представляет собой платформа ОС –

- организацию исполнения прикладных программ на уровне операционной системы.
- ☐ совокупность взаимодействующих между многопоточных программ и операционной системы
- ☐ совокупность взаимодействующих между собой кроссплатформенных библиотек и операционной системы, под управлением которой функционируют прикладные программы и средства для их разработки
- ☐ совокупность операционной системы, средств разработки прикладных программных решений и прикладных программ, работающих под управлением данной операционной системы

291. Программная платформа – это

- совокупность операционной системы, средств разработки прикладных программных решений и прикладных программ, работающих под управлением данной операционной системы.
- ☐ совокупность взаимодействующих между собой аппаратных средств и операционной системы, под управлением которой функционируют прикладные программы и средства для их разработки
- ☐ совокупность взаимодействующих между многопоточных программ и операционной системы
- ☐ совокупность взаимодействующих между собой кроссплатформенных библиотек и операционной системы, под управлением которой функционируют прикладные программы и средства для их разработки

292. Объектно-ориентированное программирование –

- подход к программированию, при котором основными концепциями являются понятия объектов и классов.
- ☐ тип данных, описывающий структуру и поведение объектов
- ☐ экземпляр класса
- ☐ элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса

293. Что такое класс –

- тип данных, описывающий структуру и поведение объектов.
- ☐ подход к программированию, при котором основными концепциями являются понятия объектов и классов
- ☐ экземпляр класса
- ☐ элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса

294. Что такое объект –

- экземпляр класса.

- ☐ тип данных, описывающий структуру и поведение объектов
 - ☐ подход к программированию, при котором основными концепциями являются понятия объектов и классов
 - ☐ элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса
295. Что такое поле –
- элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса.
 - ☐ экземпляр класса
 - ☐ элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса
 - ☐ тип данных, описывающий структуру и поведение объектов
296. Что такое состояние объекта –
- набор текущих значений полей объекта.
 - ☐ экземпляр класса
 - ☐ элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса
 - ☐ подход к программированию, при котором основными концепциями являются понятия объектов и классов
297. Что такое метод –
- процедура или функция, выполняющаяся в контексте объекта, для которого она вызывается.
 - ☐ элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса
 - ☐ тип данных, описывающий структуру и поведение объектов
 - ☐ подход к программированию, при котором основными концепциями являются понятия объектов и классов
298. Что такое свойство –
- специальный вид методов, предназначенный для модификации отдельных полей объекта.
 - ☐ подход к программированию, при котором основными концепциями являются понятия объектов и классов
 - ☐ процедура или функция, выполняющаяся в контексте объекта, для которого она вызывается
 - ☐ элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса
299. Что такое член класса –
- поля, методы и свойства класса
 - ☐ элемент данных класса
 - ☐ специальный вид методов
 - ☐ тип данных
300. Что такое модификатор доступа –
- дополнительная характеристика членов класса, определяющая, имеется ли к ним доступ из внешней программы, или же они используются исключительно в границах класса и скрыты от окружающего мира.
 - ☐ специальный вид методов, предназначенный для модификации отдельных полей объекта

- ☐ подход к программированию, при котором основными концепциями являются понятия объектов и классов
 - ☐ тип данных, описывающий структуру и поведение объектов
301. Что такое конструктор –
- специальный метод, выполняемый сразу же после создания экземпляра класса.
- ☐ подход к программированию, при котором основными концепциями являются понятия объектов и классов
 - ☐ специальный вид методов, предназначенный для модификации отдельных полей объекта
 - ☐ процедура или функция, выполняющаяся в контексте объекта, для которого она вызывается
302. К чему приводит, когда конструктор инициализирует поля объекта –
- приводит объект в начальное состояние.
- ☐ приводит класс в начальное состояние
 - ☐ приводит функцию в начальное состояние
 - ☐ приводит метод в начальное состояние
303. Как называют конструктор без параметров
- конструктор по умолчанию.
- ☐ именованный конструктор
 - ☐ конструктор копирования
 - ☐ виртуальный конструктор

Критерии оценки за пройденный итоговый тест:

- 40 баллов выставляется обучающемуся, если он ответил правильно на все вопросы случайной выборки 40 тестовых заданий;
- 0-39 баллов выставляется обучающемуся в зависимости от количества верных ответов на вопросы случайной выборки 40 тестовых заданий.

7.2.2 Примеры практических работ

Модуль 4. Технологии конструирования программного обеспечения

Практическая работа № 1

Цель работы

Сформировать навыки развертывания среды разработки веб-приложения.

Задание. Развертывание среды разработки и реализации.

Рекомендации по выполнению задания

Для выполнения работы рекомендуется установить в виртуальной машине (например, VirtualBox) операционную систему Xubuntu версии 16.04 или 18.04. Для Windows-систем установка данных серверов проходит без особых сложностей, но требует больше ресурсов компьютера. Кроме того, могут отсутствовать подробные описания по установке.

Для выполнения данной работы необходимо установить требуемые для разработки и реализации программные продукты:

1. Виртуальная машина Java JRE and JDK. Ее можно получить по ссылке: <https://www.oracle.com/technetwork/java/javase/downloads/index.html>.
2. GlassFish Application Server. Этот сервер приложений необходим для использования возможностей технологии Java EE. В качестве сервера приложений можно использовать и другие, поддерживающие технологию EJB: например, WildFly (JBoss), WebLogic или WebSphere. GlassFish Application Server можно приобрести по данному адресу: <http://glassfish.java.net/public/downloadsindex.html>.
3. SQL-сервер — MySQL Server. В качестве SQL-сервера можно также использовать любой, но с наличием драйвера JDBC для подключения с сервера приложений к SQL-серверу. MySQL Server можно получить по ссылке: <https://www.mysql.com/downloads/mysql/>.

NetBeans — среда разработки на языке Java. Это свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, JavaFX, Python, PHP, JavaScript, C++, Ада и др. NetBeans IDE можно получить по ссылке: <https://netbeans.org/downloads/index.html>.

Модуль 4. Технологии конструирования программного обеспечения

Практическая работа № 2

Цель работы

Сформировать навыки создания простейшей веб-страницы на основе технологии JSP.

Задание. Разработка веб-приложения.

Рекомендации по выполнению задания

1. Необходимо произвольно выбрать вариант задания для определения предметной области по таблице, представленной ниже (допускается по согласованию с руководителем выбрать другую тему).

№ варианта	Предметная область
1	Районная библиотека
2	Аптечный склад
3	Диспетчерская автобусного парка
4	Железнодорожная касса
5	Пункт проката велосипедов
6	Приемная комиссия вуза
7	Оптовая база товаров бытовой химии
8	Регистратура поликлиники
9	Детская библиотека
10	Салон красоты (парикмахерская)

2. Необходимо разработать веб-страницы на основе технологии JSP с возможностью чтения, добавления и редактирования выбранных из предметной области данных. Для это сначала разрабатывается структура страниц. Например, можно составить структуру: *главная страница сайта* содержит список страниц и стартовых данных; *страница чтения и редактирования данных* — полное содержание по выбранному объекту.
3. При работе с JSP необходимо реализовать бизнес-логику на основе паттерна программирования MVC (<http://rsdn.org/article/patterns/generic-mvc.xml>).
 - Модель (*M*, *Model*) хранит в себе бизнес-логику приложения, регулирует доступ к данным и их изменение.

- *Вид (V, View)* отображает содержимое модели, определяют, как необходимо представить данные, полученные от модели.
- *Контроллер (C, Controller)* определяет поведение всего приложения, получает от вида (View) пользовательские данные, интерпретирует их в действия, выполняемые с помощью модели.

Применительно к Java EE шаблон MVC реализуется на следующей основе: сервлет используется как контроллер для обработки входящих запросов пользователей от представления, которое реализуется с помощью страниц JSP. Модель представляет собой EJB — сессионные компоненты, а также классы сущности (JPA), которые, в свою очередь, содержат данные из БД (рис. 1).

Пример реализации JSP-страниц можно изучить на сайтах корпорации Oracle. На сайтах, посвященных программированию, также есть множество примеров, на которые можно опираться при выполнении задания. Например: <https://onedeveloper.javadev.ru/article%3Fid=4.html>.

4. В качестве отчета необходимо представить:

- текстовый документ с подробным алгоритмом (порядком) установки и настройки необходимых для выполнения последующих работ программных продуктов. При этом рекомендуется сделать упор на написание команд по установке, описание выполняемых настроек и сделать минимальной вставку скриншотов;
- текстовый документ с описанием порядка разработки и развертывания страниц JSP на сервере приложений;
- исходный и скомпилированный код разработанных JSP-страниц.

Отчет и исходный код следует разместить на сайте системы управления репозиториями кода Git: <https://gitlab.com>. Для оценивания нужно указать адрес страницы практики на сайте gitlab.com.

Модуль 4. Технологии конструирования программного обеспечения

Практическая работа № 3

Цель работы

Сформировать навыки разработки и реализации веб-приложения.

Задание. Разработка веб-приложения. Создание простейшего сервлета.

Рекомендации по выполнению задания

1. На основе предыдущего задания необходимо разработать дескриптор развертывания и JSP-сегменты. Дескриптор развертывания — это файл XML-файл, который описывает настройки развертывания компонента на сервере. Приложение Java EE и каждый его модуль содержит свой дескриптор. Данные в дескрипторе объявляются декларативно и могут быть изменены без вмешательства в исходный код. Сервер приложений Java EE выполняет запуск веб-приложения по алгоритму, указанному в дескрипторе развертывания.

JSP-сегменты необходимы для создания на страницах отдельных областей (фреймов), в которых будет находиться повторяющаяся часть кода. К ним относятся чаще всего заголовок (header) и строка состояния (footer). Код располагается в отдельных файлах. В технологии JSP эти файлы располагаются в отдельном каталоге JSPF в папке WEB-INF. Расширение файлов также будет .jspx.

2. Необходимо разработать сервлет для реализации динамической обработки данных. Сервлет — это Java-интерфейс, позволяющий расширить функционал веб-сервера. Сервлет принимает и обрабатывает запросы, поступившие от пользователя. Затем он определяет, как поступить дальше и какой ответ дать пользователю. Для разных

запросов пользователя можно назначить свой сервлет. С описанием процесса разработки сервлета можно ознакомиться по ссылкам: <http://java-online.ru/servlet.shtml> и <https://onedevolver.javadev.ru/article%3Fid=5.html>.

3. В качестве отчета необходимо представить:

- текстовый документ с описанием порядка разработки и развертывания сервлета на сервере приложений;
- исходный и скомпилированный код разработанного сервлета;

Отчет и исходный код следует разместить на сайте системы управления репозиториями кода Git: <https://gitlab.com>. Для оценивания нужно указать адрес страницы практики на сайте gitlab.com.

Модуль 4. Технологии конструирования программного обеспечения

Практическая работа № 4

Тема 4. Часть 3. Языки конструирования

Цель работы

Сформировать навыки разработки и реализации веб-приложения.

Задание. Разработка веб-приложения с использованием технологий EJB и JPA

Рекомендации по выполнению задания

1. На основе предыдущего задания необходимо разработать, реализовать и наполнить минимальным содержимым базу данных создаваемого вами веб-приложения.
2. Требуется создать JDBC-соединение и настроить пул соединений (Connecting pool) с базой данных на установленном сервере приложений. Пример настройки можно изучить по адресу: <http://onedevolver.ru/article?id=6>.
3. Нужно разработать JPA-контейнер для разработанных и реализованных выше сущностей на SQL-сервере. JPA-контейнер выполняет объектно-реляционное отображение простых Java-объектов на сущности, хранимые на SQL-сервере. При этом JPA-контейнер предоставляет API для сохранения, получения и управления такими объектами. С примером реализации JPA контейнера можно ознакомиться по ссылке: <http://onedevolver.ru/article?id=7>.
4. Требуется разработать сессионный компонент и EJB-контейнер для отображения выбранных сведений из реализованной базы данных на разработанных JSP-страницах. Пример реализации EJB-контейнера представлен на сайте <http://onedevolver.ru/article?id=7>.
5. В качестве отчета необходимо представить:
 - текстовый документ с описанием порядка разработки и развертывания контейнеров JPA и EJB на сервере приложений;
 - исходный и скомпилированный код разработанных JPA- и EJB-контейнеров.Отчет и исходный код следует разместить на сайте системы управления репозиториями кода Git: <https://gitlab.com>. Для оценивания нужно указать адрес страницы практики на сайте gitlab.com.

Требования к оформлению

Отчет должен содержать подробное описание (включая иллюстративный материал) последовательности действий, проделанных студентом для выполнения заданий. Оформление отчета должно соответствовать методическому указанию рекомендациям, изложенным учебно-методическом пособии [Очеповский А.В. Общие требования по выполнению и оформлению контрольных, курсовых и выпускных квалификационных работ: Учебно-методическое пособие. – Тольятти: ТГУ, 2015. 78 с.].

Процедура оценивания

Оценка выполненной работы проводится по критериям:

1. Наличие всей существенной информации по работе
2. Точность и полнота предоставляемых сведений
3. Непротиворечивость приводимой информации
4. Правильность интерпретаций и выводов, которые сделаны по результатам работы
5. Степень достижения студентом поставленной цели
6. Обоснованность применяемого решения
7. Грамотность (содержательная) используемых формулировок

Критерии оценки за отчеты по практическим работам:

Формы текущего контроля	Критерии и нормы оценки
Отчеты по практическим работам 1, 2	6 баллов – задание выполнено в полном объеме без замечаний 4 балла – задание выполнено в объеме 70% без замечаний, или задание выполнено в полном объеме, но присутствуют замечания. 3 балла – задание выполнено в объеме 50% без замечаний, или задание выполнено в полном объеме, но присутствуют большое кол-во замечаний 1 балл – задание выполнено в объеме менее 50%. 0 баллов – задание не выполнено.
Отчеты по практическим работам 3,4	10 баллов – задание выполнено в полном объеме без замечаний 7 баллов – задание выполнено в объеме 70% без замечаний, или задание выполнено в полном объеме, но присутствуют замечания. 5 баллов – задание выполнено в объеме 50% без замечаний, или задание выполнено в полном объеме, но присутствуют большое кол-во замечаний 2 балла – задание выполнено в объеме менее 50%. 0 баллов – задание не выполнено.

Критерии оценки за пройденный итоговый тест:

40 баллов выставляется обучающемуся, если он ответил правильно на все вопросы случайной выборки 30 тестовых заданий;

0-39 баллов выставляется обучающемуся в зависимости от количества верных ответов на вопросы случайной выборки 30 тестовых заданий

7.3. Оценочные средства для промежуточной аттестации по итогам освоения дисциплины

7.3.1. Вопросы к промежуточной аттестации

1. Приведите понятие конструирование программного обеспечения
2. Приведите понятие Кодирование, Верификация, Валидация
3. Что такое Интеграционное тестирование и Модульное тестирование
4. Что такое Минимизация сложности ПО и Ожидание изменений при разработке ПО

5. Что такое Конструирование с возможностью проверки и Повторное использование кода
6. Перечислите Стандарты для коммуникационных методов и раскройте один из стандартов для коммуникационных методов
7. Перечислите Стандарты для языков программирования и раскройте один из стандартов для языков программирования
8. Перечислите Стандарты кодирования и раскройте один из стандартов кодирования
9. Перечислите Стандарты программных платформ и интерфейсов и раскройте один из стандартов программных платформ и интерфейсов
10. Перечислите Стандарты для реализации инструментов при проектировании ПО и раскройте один из стандартов для реализации инструментов при проектировании ПО
11. Поясните понятие Внешние или внутренние стандарты
12. Дайте классификацию инструментов разработки программного обеспечения
13. Дайте характеристику и приведите пример ассемблеров
14. Дайте характеристику и приведите пример трансляторов
15. Дайте характеристику и приведите пример компиляторов
16. Дайте характеристику и приведите пример интерпретаторов
17. Дайте характеристику и приведите пример компоновщиков (редакторы связей)
18. Дайте характеристику и приведите пример препроцессоров исходных текстов
19. Дайте характеристику и приведите пример Отладчиков (debugger)
20. Дайте характеристику и приведите пример - Специализированные редакторы исходных текстов
21. Дайте характеристику и приведите пример - Библиотеки подпрограмм
22. Дайте характеристику и приведите пример - Редакторы графического интерфейса
23. Дайте характеристику и приведите пример - Интегрированные среды разработки
24. Дайте характеристику и приведите пример - SDK (software development kit)
25. Дайте характеристику и приведите пример - Парсеры и генераторы парсеров
26. Дайте характеристику и приведите пример - Генераторы документации
27. Дайте характеристику и приведите пример - Средства анализа покрытия кода
28. Дайте характеристику и приведите пример - Средства непрерывной интеграции
29. Дайте характеристику и приведите пример - Средства автоматизированного тестирования
30. Дайте характеристику и приведите пример - Системы управления версиями
31. Что такое Интегрированная среда разработки и приведите примеры Интегрированных сред разработки
32. Структура системы программирования
33. Дайте характеристику одной из современных интегрированных сред программирования
34. Приведите понятие по модели жизненного цикла проекта (Project Life Cycle Management - PLCM)
35. Перечислите наиболее используемые стандарты по процессу разработки ПО
36. ГОСТ «Единая система программной документации» (ЕСПД)
37. SW-CMM (Capability Maturity Model for Software)
38. Унифицированный процесс (Rational Unified Process, RUP)
39. Microsoft Solutions Framework (MSF)
40. PSP/TSP (Personal Software Process / Team Software Process)
41. Agile (Agile software development, agile-методы) гибкая методология разработки
42. Разработка через тестирование (англ. test-driven development, TDD)
43. Раскройте основные постулаты, чтобы программный проект стал успешным
44. Понятие Планирование конструкторской деятельности и на чем базируется Процесс планирования конструкторской деятельности
45. Что включает Примерный план процесса планирования конструкторской деятельности
46. Что такое Метрика программного обеспечения (software metric) и перечислите классы метрик для программного кода
47. Опишите Количественные метрики

48. Опишите Метрики сложности потока управления программы
49. Опишите Метрики сложности потока управления данными
50. Опишите Метрики сложности потока управления и данных программы
51. Перечислите чаще всего используемые метрики при кодировании
52. Что такое Степень покрытия кода тестированием
53. Что такое Рефакторинг (Refactoring) и цель рефакторинга
54. Раскройте видимые проблемы в коде, требующие рефакторинга
55. Наиболее используемые методы рефакторинга и перечислите средства, позволяющие автоматизировать процесс рефакторинга
56. Что такое Простейший тип языков конструирования и перечислите Основные языки конфигурирования
57. Опишите Формат конфигурационного файла
58. Что такое Инструментальный язык (toolkit language), Сценарный язык (scripting language) и Скриптовый язык. Перечислите типы Сценарных языков
59. Основные этапы кодирования и раскройте основы кодирования
60. Что такое Стандарт оформления кода (coding standards, coding convention или programming style). Что описывает Стандарт (стиль) оформления кода
61. Раскройте способы выбора названий
62. Раскройте стиль отступов (индентация) при оформлении логических блоков
63. Раскройте стиль комментариев и использование документирующих комментариев
64. Раскройте учет различных особенностей языка
65. Раскройте стиль именования переменных, констант и функций
66. Что такое Тестирование программного обеспечения и опишите две формы тестирования
67. Что такое Модульное тестирование (unit testing) и Интеграционное тестирование (integration testing)
68. Что такое Повторное использование кода (code reuse) и раскройте Задачи, связанные с повторным использованием в процессе конструирования
69. Что такое Динамические библиотеки и Статические библиотеки
70. Перечислите Критерии качества кода
71. Раскройте основные техники обеспечения качества
72. Перечислите типы интеграций отдельно сконструированных процедур, классов, компонентов и подсистем (модулей) в единую систему и приведите критерий интегрируемости
73. Что такое Шаблоны (template) в программировании
74. Что такое Обобщённое программирование (generic programming)
75. Что такое Отказоустойчивость
76. Опишите Жизненный цикл ошибок в ПО
77. Что такое Баг (bug) и что такое защитное программирование и его принципы
78. Что такое GIGO (Garbage In, Garbage Out)
79. Раскройте принципы хорошей программы
80. Раскройте Способы обработки входных мусорных данных
81. Раскройте в защитном программировании использование утверждения и принципы использования утверждений
82. Раскройте подходы для обработки возможных ошибок
83. Что такое обработка исключительных ситуаций (exception handling) и приведите примеры исключительных ситуаций
84. Механизмы функционирования обработчиков исключений
85. Раскройте Сущность проверяемых исключений
86. Раскройте изоляцию повреждений (баррикада)
87. Что такое MDD, MDA, xUML и приведите стандарты, на которые опирается ядро MDA
88. Что такое Автоматное программирование и раскройте особенности автоматного программирования. Приведите пример автоматного программирования

89. Что такое Интернационализация программного обеспечения. В чем различие интернационализации и локализации программного обеспечения
90. Раскройте три уровня локализации программного обеспечения
91. Приемы для интернационализации программного обеспечения
92. Раскройте использование идентификатора для установки локали (locale)
93. Что такое Middleware — промежуточное программное обеспечение и приведите классификацию Middleware
94. Что такое Профилирование ПО и приведите Примеры профилирующих программ

7.3.2 Критерии и нормы оценки

Семестр	Форма проведения промежуточной аттестации	Критерии и нормы оценки	
3	Экзамен (по накопительному рейтингу)	отлично	От 80 до 100 баллов
		хорошо	От 60 до 79 баллов
		удовлетворительно	От 40 до 59 баллов
		неудовлетворительно	менее 40 баллов

8. Учебно-методическое и информационное обеспечение дисциплины

8.1 Обязательная литература

№ п/п	Авторы, составители	Заглавие (заголовок)	Тип (учебник, учебное пособие, учебно-методическое пособие, практикум, др.)	Год издания	Количество в научной библиотеке / Наименование ЭБС
1		Администрирование ОС Unix [Электронный ресурс] : [курс лекций]. - 2-е изд., испр. - Москва : ИНТУИТ, 2016. - 304 с. : ил.	Учебное пособие	2016	ЭБС «IPRbooks»
2		Введение в программные системы и их разработку [Электронный ресурс] : [учеб. пособие] / С. В. Назаров [и др.]. - 2-е изд., испр. - Москва : ИНТУИТ, 2016. - 649с.	Учебное пособие	2016	ЭБС «IPRbooks»
3		Кознов Д. В. Введение в программную инженерию [Электронный ресурс] : [учебное пособие] / Д. В. Кознов. - Москва : ИНТУИТ, 2016. - 306 с. : ил.	Учебное пособие	2016	ЭБС «IPRbooks»
4		Липаев В. В. Программная инженерия сложных заказных программных продуктов [Электронный ресурс] : учеб. пособие / В. В. Липаев. - Москва : МАКС Пресс, 2014. - 312 с. - ISBN 978-5-317-04750-4 .	Учебное пособие	2014	ЭБС «IPRbooks»
5		Операционная система UNIX [Электронный ресурс] : [учебное пособие] / Г. В. Курячий. - 2-е изд., испр. - Москва : ИНТУИТ, 2016. - 258 с. : ил. - ISBN 5-9556-0019-1.	Учебное пособие	2016	ЭБС «IPRbooks»
6		Основы автоматизированного проектирования [Электронный ресурс] : учебник / под ред. А. П. Карпенко . - Москва : ИНФРА-М, 2015. - 329 с. : ил. - (Высшее образование. Бакалавриат). - ISBN 978-5-16-010213-9.	Учебное пособие	2015	ЭБС «Znanium.com»

8.2 Дополнительная литература

№ п/п	Авторы, со- ставители	Заглавие (заголовок)	Тип (учебник, учебное пособие, учебно- методическое по- собие, практи- кум, др.)	Год из- дания	Количество в научной биб- лиотеке / Наименова- ние ЭБС
1		Антамошкин О. А. Программная инженерия [Электронный ресурс] : теория и практика : учеб. / О. А. Антамошкин ; Сибирский федеральный университет. - Красноярск : СФУ, 2012. - 247 с.	Учебное пособие	2000	0
2		Гибридные адаптивные интеллектуальные системы. Часть 1. Теория и технология разработки [Электронный ресурс]: монография/ П.М. Клячек [и др.]. – Калининград: Балтийский федеральный университет им. Иммануила Канта, 2011. – 374с. ISBN 978-5-9971-0140-4	Учебное пособие	2011	ЭБС «IPRbooks»
3		Керниган Б. В. UNIX - универсальная среда программирования = The UNIX programming environment / Б. В. Керниган, Р. Пайк ; пер. с англ. А. М. Березко, В. А. Иващенко; под ред. М. И. Белякова. - Москва : Финансы и статистика, 1992. - 303 с. - Предм. указ.: с. 299-301.	Учебное пособие	1992	34
4		Командная строка UNIX [Электронный ресурс] : лабораторный практикум по дисциплине «Операционные системы». - Москва : МГСУ, 2013. - 44 с. : ил.	Учебное пособие	2013	ЭБС «IPRbooks»
5		Королева О. Н. Базы данных [Электронный ре-сурс] : курс лекций / О. Н. Королева, Т. В. Королева, А. В. Мажукин ; ред. В. И. Мажукин. - 2-е изд., испр. и доп. - Москва : МосГУ, 2012. - 66 с. : ил. - (Информационные системы и технологии в экономике и управлении). - ISBN 978-5-98079-838-3.	Учебное пособие	2012	ЭБС «IPRbooks»
6		Котляров В. П. Основы тестирования программного обеспечения [Электронный ресурс] : учеб. пособие / В. П. Котляров, Т. В.	Учебное пособие	2013	ЭБС «IPRbooks»

№ п/п	Авторы, со- ставители	Заглавие (заголовок)	Тип (учебник, учебное пособие, учебно- методическое по- собие, практи- кум, др.)	Год из- дания	Количество в научной биб- лиотеке / Наименова- ние ЭБС
		Коликова. - Москва : БИНОМ : Лаборатория знаний : ИНТУИТ, 2013. - 285 : ил. - (Основы информационных технологий). - ISBN 5-94774-406-4.			
7		Курячий Г. В. Операционная система Linux [Электронный ресурс] : курс лекций : учеб. пособие / Г. В. Курячий, К. А. Маслинский. – Саратов : Профобразование, 2017. - 347 с. : ил. - ISBN 978-5-4488-0110-5.	Учебное пособие	2017	ЭБС «IPRbooks»
8		Мошков М. Е. Введение в системное администрирование Unix [Электронный ресурс] : [курс лекций] / М. Е. Мошков. - 2-е изд., испр. - Москва : ИНТУИТ, 2016. - 209 с. : ил.	Учебное пособие	2016	ЭБС «IPRbooks»
9		Муратова С. Ю. Макросы и приложения [Электронный ресурс] : лабораторный практикум / С. Ю. Муратова. - Москва : МИ-СиС, 2013. - 152 с. - ISBN 978-5-87623-716-3.	Учебное пособие	2013	ЭБС «Лань»
10		Назаров С. В. Современные операционные системы [Электронный ресурс] : [учеб. пособие] / С. В. Назаров, А. И. Широков. - 2-е изд., испр. - Москва : ИНТУИТ, 2016. - 351 с. : ил. - (Основы информационных технологий). - ISBN 978-5-9963-0416-5.	Учебное пособие	2016	ЭБС «IPRbooks»

8.3 Перечень профессиональных баз данных и информационных справочных систем

1. About SWEBOOK. Режим доступа: <https://www.computer.org/web/swebok>, 2022-01-01.
2. Java и вы. Режим доступа: <http://www.java.com/ru/>, 2022-01-01.
3. Oracle Technology Network - Java. Режим доступа: <http://www.oracle.com/technetwork/java/index.html>, 2022-01-01.
4. Project Management Institute. Режим доступа: <http://www.pmi.org/>, 2022-01-01.

8.4 Перечень программного обеспечения

№ п/п	Наименование ПО	Количество лицензий	Реквизиты договора (дата, номер, срок действия)
1	Eclipse Foundation Eclipse версия 4	неограни- ченный	Лицензия Eclipse Public License
2	NetBeans Community NetBeans IDE версия 8	неограничен ный	Лицензия LGPLv2.1, GPLv2 with Classpath exception
3	The CodeBlocks team CodeBlocks версия 16	неограничен ный	Лицензия GNU GPLv3