

Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Щёкина Вера Витальевна
Должность: Ректор
Дата подписания: 22.10.2026 03:51:30
Уникальный программный ключ:
a2232a55157e578551a8999b11908926539894704701100157a454e17789

	МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
	Федеральное государственное бюджетное образовательное учреждение высшего образования «Благовещенский государственный педагогический университет»
	ОСНОВНАЯ ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА Рабочая программа дисциплины

УТВЕРЖДАЮ

Декан

физико-математического факультета

ФГБОУ ВО «БГПУ»

 Т.А. Мередилина
«24» мая 2023 г.

Рабочая программа дисциплины
ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ JAVA

Направление подготовки

02.03.03 МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ И
АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Профиль

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

Уровень высшего образования
БАКАЛАВРИАТ

Принята
на заседании кафедры информатики
и методики преподавания информатики
(протокол № 8 от «24» мая 2023 г.)

Благовещенск 2023

СОДЕРЖАНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА	3
2 УЧЕБНО-ТЕМАТИЧЕСКОЕ ПЛАНИРОВАНИЕ	5
3 СОДЕРЖАНИЕ ТЕМ (РАЗДЕЛОВ)	6
4 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ (УКАЗАНИЯ) ДЛЯ СТУДЕНТОВ ПО ИЗУЧЕНИЮ ДИСЦИПЛИНЫ	7
5 ПРАКТИКУМ ПО ДИСЦИПЛИНЕ	11
6 ДИДАКТИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ КОНТРОЛЯ (САМОКОНТРОЛЯ) УСВОЕННОГО МАТЕРИАЛА.....	14
7 ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ	43
В ПРОЦЕССЕ ОБУЧЕНИЯ	43
8 ОСОБЕННОСТИ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ ИНВАЛИДАМИ И ЛИЦАМИ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ	44
9 СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ	44
10 МАТЕРИАЛЬНО-ТЕХНИЧЕСКАЯ БАЗА	45
11 ЛИСТ ИЗМЕНЕНИЙ И ДОПОЛНЕНИЙ	46

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

1.1 Цель дисциплины: формирование у студентов представления о современных подходах и методах программирования на примере наиболее популярного языка программирования Java и сопряженных технологий.

1.2 Место дисциплины в структуре ООП: Дисциплина «Технология программирования Java» относится к дисциплинам обязательной части блока Б1 (Б1.О.28).

1.3 Дисциплина направлена на формирование следующих компетенций: ОПК-3, ПК-4, ПК-5, ПК-7.

- **ОПК-3.** Способен понимать и применять современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения, **индикаторами** достижения которой является:

- ОПК-3.1 – **знать** основные положения и концепции прикладного и системного программирования, архитектуры компьютеров и сетей (в том числе и глобальных), современные языки программирования, технологии создания и эксплуатации программных продуктов и программных комплексов;

- ОПК-3.2 – **уметь** использовать их в профессиональной деятельности;

- ОПК-3.3 – **иметь практические навыки** разработки программного обеспечения.

- **ПК-4.** Способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях, **индикаторами** достижения которой является:

- ПК-4.1 – **знать** современные технологии проектирования и производства программного продукта;

- ПК-4.2 – **уметь** использовать подобные технологии при создании программных продуктов;

- ПК-4.3 – **иметь практический опыт** применения подобных технологий.

- **ПК-5.** Способен использовать основные методы и средства автоматизации проектирования, реализации, испытаний и оценки качества при создании конкурентоспособного программного продукта и программных комплексов, а также способен использовать методы и средства автоматизации, связанные с сопровождением, администрированием и модернизацией программных продуктов и программных комплексов., **индикаторами** достижения которой является:

- ПК-5.1 – **знать** современные приемы работы с инструментальными средствами, поддерживающими создание программных продуктов и программных комплексов, их сопровождения и администрирования;

- ПК-5.2 – **уметь** использовать подобные инструментальные средства в практической деятельности;

- ПК-5.3 – **иметь практический опыт** применения подобных инструментальных средств.

- **ПК-7.** Способен использовать основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы, способы и средства разработки программ в рамках этих направлений, **индикаторами** достижения которой является:

- ПК-7.1 – **знать** основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы, способы и средства разработки программ в рамках этих направлений;

- ПК-7.2 – **уметь** программировать в рамках этих направлений;
- ПК-7.3 – **иметь практический опыт** разработки программ в рамках этих направлений.

1.4 Перечень планируемых результатов обучения. В результате изучения дисциплины студент должен

знать:

- ключевые парадигмы современного программирования;
- необходимые инструментальные средства для разработки современных программных приложений;
- основы языка программирования Java;

структуру основных пакетов классов, используемых в большинстве разработок на Java;

уметь:

- решать базовые задачи автоматизации;
- уметь объяснить способы и методы решения;
- самостоятельно изучать на существующих примерах программного кода способы разработки программного обеспечения;

владеть:

- базовыми алгоритмами и техниками решения учебных задач программирования современными языковыми и технологическими средствами.

1.5 Общая трудоемкость дисциплины «Технология программирования Java» составляет 5 зачетных единиц (далее – ЗЕ) (180 часов).

Программа предусматривает изучение материала на лекциях и практических занятиях. Предусмотрена самостоятельная работа студентов по темам и разделам. Проверка знаний осуществляется фронтально, индивидуально.

1.6 Объем дисциплины и виды учебной деятельности

Объем дисциплины и виды учебной деятельности (очная форма обучения)

Вид учебной работы	Всего часов	Семестр 5
Общая трудоемкость	180	180
Аудиторные занятия	80	80
Лекции	32	32
Лабораторные работы	48	48
Самостоятельная работа	64	64
Вид итогового контроля	36	экзамен

2 УЧЕБНО-ТЕМАТИЧЕСКОЕ ПЛАНИРОВАНИЕ

2.1 Очная форма обучения

Учебно-тематический план

№	Наименование тем (разделов)	Всего часов	Аудиторные занятия		Самостоятельная работа
			Лекции	Лабораторные работы	
1.	Введение в технологию Java	6	2		4
2.	Ключевые свойства технологии Java	6	2		4
3.	Объектно-ориентированный подход к разработке ПО	6	2		4
4.	Виды классов, интерфейсы, пакеты	10	2	4	4
5.	Вывод и ввод информации	8	2	2	4
6.	Примитивные типы данных	8	2	2	4
7.	Массивы одномерные и многомерные	10	2	4	4
8.	Средства управления ходом выполнения программы: следования, ветвления, циклы	10	2	4	4
9.	Динамические структуры данных средствами Java	10	2	4	4
10.	Работа с файлами, потоки ввода-вывода	10	2	4	4
11.	Создание графического интерфейса программ: введение	8	2	2	4
12.	Создание графического интерфейса программ: менеджеры компоновки	14	2	6	6
13.	Создание графического интерфейса программ: списки, таблицы, редакторы, сложные компоненты	14	2	6	6
14.	Обработка строк	8	2	4	2
15.	Сетевые средства языка Java	14	2	6	6
16.	Перспективы развития технологии Java	2	2	-	-
Экзамен		36			
ИТОГО		180	32	48	64

Интерактивное обучение по дисциплине

№	Наименование тем (разделов)	Вид занятия	Форма интерактивного занятия	Кол-во часов
1.	Вывод и ввод информации	Лк	Проблемная лекция	2
2.	Массивы одномерные и многомерные	Лк	Проблемная лекция	2
3.	Средства управления ходом выполнения программы: следования, ветвления, циклы	Лк	Проблемная лекция	2
4.	Работа с файлами, потоки ввода-вывода	Лк	Проблемная лекция	2
5.	Создание графического интерфейса программ: введение	Лк	Проблемная лекция	2
6.	Массивы одномерные и многомерные	Лб	Проблемная лабораторная работа	2
7.	Средства управления ходом выполнения программы: следования, ветвления, циклы	Лб	Проблемная лабораторная работа	2
8.	Работа с файлами, потоки ввода-вывода	Лб	Проблемная лабораторная работа	2
9.	Создание графического интерфейса программ: введение	Лб	Проблемная лабораторная работа	4
10.	Создание графического интерфейса программ: менеджеры компоновки	Лб	Проблемная лабораторная работа	4
ИТОГО				24

3 СОДЕРЖАНИЕ ТЕМ (РАЗДЕЛОВ)

Тема 1. Введение в технологию Java.

Эволюция технологий, приведших к появлению платформы Java и оказавших дальнейшее влияние на ее развитие.

Тема 2. Ключевые свойства технологии Java.

Обзор ключевых свойств технологии, таких как мобильность, устойчивость к ошибкам и легкость их обработки, объектная ориентированность, поддержка сетей на языковом уровне, поддержка распределенных вычислений, интерпретируемость в сочетании с компилируемостью, безопасность.

Тема 3. Объектно-ориентированный подход к разработке ПО.

Существо объектно-ориентированной парадигмы программирования. Понятия инкапсуляции, наследования и полиморфизма.

Тема 4. Виды классов, интерфейсы, пакеты.

Реализация ООП в языке программирования Java. Виды классов. Интерфейсы. Пакеты. Импорт пакетов, создание пакетов.

Тема 5. Вывод и ввод информации.

Разбор конкретных примеров организации ввода информации с клавиатуры и вывода ее на консоль.

Тема 6. Примитивные типы данных.

Детальное рассмотрение восьми примитивных типов данных в Java. Примеры их применения.

Тема 7. Массивы одномерные и многомерные.

Создание одномерных и многомерных массивов произвольной размерности. Демонстрация принципов выделения памяти виртуальной машиной, а также обращения к объектам по ссылке-имени. Заполнение массивов, обход массива, вывод массива на экран.

Тема 8. Средства управления ходом выполнения программы: следования, ветвления, циклы.

Три вида циклов в Java. Оператор ветвления if..else, switch. Команды break, continue.

Тема 9. Динамические структуры данных средствами Java.

Пакет java.util и классы динамических структур. Реализация списков, очередей, стеков, деревьев, множеств.

Тема 10. Работа с файлами, потоки ввода-вывода.

Работа с потоками ввода-вывода на примере работы с текстовыми файлами. Организация записи данных в файл и чтения из файла с выводом на экран.

Тема 11. Создание графического интерфейса программ: введение.

Проектирование простого интерфейса с помощью редактора форм Matisse в NetBeans. Самостоятельное создание форм, кнопок, меток и других управляющих элементов формы.

Тема 12. Создание графического интерфейса программ: менеджеры компоновки.

Разбор стандартных менеджеров компоновки, их назначения и методов реализации.

Тема 13. Создание графического интерфейса программ: списки, таблицы, редакторы, сложные компоненты.

Демонстрация создания и использования сложных компонентов javax.swing.

Тема 14. Обработка строк.

Методы и практические задачи обработки строк.

Тема 15. Сетевые средства языка Java.

Создание клиентской и серверной части программного обеспечения. Загрузка содержимого из Интернет.

Тема 16. Перспективы развития технологии java

Изучение возможных направлений развития технологии Java

4 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ (УКАЗАНИЯ) ДЛЯ СТУДЕНТОВ ПО ИЗУЧЕНИЮ ДИСЦИПЛИНЫ

4.1 Общие методические рекомендации

В ходе изучения дисциплины достигается освоение студентами теории и практики разработки приложений на Java, формируется представление и навык реализации на современном языке программирования наиболее актуальных алгоритмов, навыки разработки приложений в современных интегрированных средах разработки.

Представленные материалы призваны организовать процесс изучения дисциплины «Технология программирования Java».

Список литературы позволяет использовать материалы как для подготовки к лабораторным работам, так и для организации самостоятельной подготовки, а также расширения представлений о направлениях исследований, так или иначе связанных с дисциплиной.

4.2 Методические рекомендации по подготовке к лекциям

Приступая к изучению дисциплины «Технология программирования Java» студент должен иметь представление о современных подходах к разработке программного обеспечения, о практическом применении получаемых в рамках курса знаний.

Самостоятельная подготовка к лекциям происходит до посещения занятий путем чтения рекомендованной литературы и выполнения задач, полученных на практических занятиях. Выполнение многих задач требует больше времени, чем отведено для работы в аудиториях университета.

Посещение лекции и активное участие в интерактивных формах обучения является еще одной формой самостоятельной работы студента. Конспектирование ключевых мыслей и программного кода не является обязательным компонентом такой работы, но рекомендуется, так как повышает эффективность выполнения заданий в ходе практических занятий.

Важной частью самостоятельной работы является периодическое повторение пройденного материала, что способствует более глубокому усвоению знаний и упрощает продвижение по пути освоения последующих тем.

4.3 Методические рекомендации по подготовке к лабораторным занятиям

Целью лабораторных занятий является закрепление теоретического материала лекций и выработка умения использования информационных и других ресурсов, предоставляемых университетом.

Подготовка к лабораторным работам предполагает изучение теоретического материала по указанной теме, с использованием конспектов лекций и дополнительной литературы. При необходимости можно обращаться за консультацией к преподавателю.

В процессе подготовки к занятиям рекомендуется взаимное обсуждение материала, во время которого закрепляются знания, а также приобретает практика в изложении и разъяснении полученных знаний, развивается речь.

В случае появления каких-либо вопросов следует обращаться к преподавателю в часы его консультаций.

Для проведения практических занятий используются компьютеры, оснащенные ОС Windows XP и выше, ОС Linux, Java Development Kit, NetBeans, система электронного образования университета. Возможно использование проектора или интерактивной доски.

4.4 Методические рекомендации к самостоятельной работе студентов

Самостоятельная работа студентов при изучении дисциплины «Технология программирования Java» организуется с целью формирования профессиональных компетенций, понимаемых как способность применять знания, умения и личностные качества для успешной деятельности в определенной области, в том числе:

- формирования умений по поиску и использованию различных источников информации;
- качественного освоения и систематизации полученных теоретических знаний, их углубления и расширения по применению на уровне межпредметных связей;
- формирования умения применять полученные знания на практике;
- развития познавательных способностей студентов, формирования самостоятельности мышления;
- развития активности, творческой инициативы, самостоятельности, ответственности и организованности;
- формирования способностей к саморазвитию;

В ходе изучения дисциплины «Технология программирования Java» предлагается выполнить различные виды самостоятельной работы:

- выполнение индивидуальных заданий лабораторных работ;

- подготовка к аудиторным занятиям;
- изучение отдельных тем (вопросов) дисциплины в соответствии с учебно-тематическим планом, составление конспектов;
- подготовка ко всем видам контрольных испытаний.

В методических указаниях излагается порядок выполнения лабораторных работ. При выполнении работ используются Java Development Kit, NetBeans.

К зачету по лабораторной работе предъявляется архивный файл NetBeans-проекта, сохраненный в своем личном кабинете В СЭО БГПУ.

4.5 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине:

1. Оценочные средства.
2. Задания.
3. Список тем для собеседования на экзамене.
4. Список литературы и информационных ресурсов.

Учебно-методическое обеспечение самостоятельной работы студентов по дисциплине

Очная форма обучения

№	Наименование раздела (темы)	Формы/виды самостоятельной работы	Количество часов, в соответствии с учебно-тематическим планом
1.	Введение в технологию Java	Проработка теоретического материала по конспектам лекций	4
2.	Ключевые свойства технологии Java	Проработка теоретического материала по конспектам лекций	4
3.	Объектно-ориентированный подход к разработке ПО	Проработка теоретического материала по конспектам лекций	4
4.	Виды классов, интерфейсы, пакеты	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	4
5.	Вывод и ввод информации	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	4
6.	Примитивные типы данных	Проработка теоретического материала по конспектам лекций. Решение задач.	4

		Подготовка отчетов о выполнении лабораторных работ.	
7.	Массивы одномерные и многомерные	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	4
8.	Средства управления ходом выполнения программы: следования, ветвления, циклы	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	4
9.	Динамические структуры данных средствами Java	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	4
10.	Работа с файлами, потоки ввода-вывода	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	4
11.	Создание графического интерфейса программ: введение	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	4
12.	Создание графического интерфейса программ: менеджеры компоновки	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	6
13.	Создание графического интерфейса программ: списки, таблицы, редакторы, сложные компоненты	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ	6
14.	Обработка строк	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	2

15.	Сетевые средства языка Java	Проработка теоретического материала по конспектам лекций. Решение задач. Подготовка отчетов о выполнении лабораторных работ.	6
	ИТОГО		64

5 ПРАКТИКУМ ПО ДИСЦИПЛИНЕ

Тема 1. Решение простых вычислительных задач

Содержание

- Вычисление факториала числа, введенного с клавиатуры.
- Выяснить, является ли произвольно введенное число простым.
- Вывести все простые числа в диапазоне до N

Литература:

Шилдт Г. Java. Наиболее полное руководство, 8-е изд. – М.: ООО «ИД Вильямс», 2012. – 1104 с.

Семочкин, А.Н. Язык программирования Java : учеб. пособие для студ. вузов / А. Н. Семочкин ; М-во образования и науки Рос. Федерации, Федеральное агентство по образованию [и др.]. – 2-е изд., перераб. и доп. – Благовещенск : Изд-во БГПУ, 2006, 2009. – 89 с. (12)

Тема 2. Создание классов, интерфейсов и пакетов

Содержание

Построить иерархию классов, абстрактных классов, интерфейсов для условной структуры «Зоопарк», где виды животных определяются абстрактными классами, конкретные представители конкретными классами, а способы их взаимодействия определяются интерфейсами. Реализовать сумму простых реакций животных на действия друг друга: ходить, рычать, есть, спать и тому подобное.

Литература:

Ноутон, П. Java 2 : перевод с английского / П. Ноутон, Г. Шилдт. – Дюссельдорф; Киев ; М. ; СПб. : БхВ – Петербург, 2001. – 1050 с. (10)

Эккель Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

Тема 3. Ввод и вывод информации

Содержание

Написание диалоговой программы в виде теста в ходе выполнения лабораторной работы.

Литература:

Ноутон, П. Java 2 : перевод с английского / П. Ноутон, Г. Шилдт. – Дюссельдорф; Киев ; М. ; СПб. : БхВ – Петербург, 2001. – 1050 с. (10)

Эккель Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

Тема 4. Массивы

Содержание

- заполнение массива случайными значениями и вывод на экран.
- сортировка массива пузырьковым методом.
- создание двумерного массива произвольных размерностей.
- поворот двумерной матрицы на 90, 180 градусов
- зеркальное отражение матрицы
- реализация класса «Список» на основе массивов с возможностью динамического расширения.

Литература:

1. Шилдт Г. Java. Наиболее полное руководство, 8-е изд. – М.: ООО «ИД Вильямс», 2012. – 1104 с.
2. Эккель Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

Тема 5. Динамические структуры данных

Содержание

- создание списков ArrayList, LinkedList.
- создание модели колоды карт на основе классов, реализующих интерфейс Set
- Написать консольную программу, которая бы по каждой букве алфавита выводила список слов, начинающихся с этой буквы. Слова содержатся в тексте, считанном из указанного файла. В качестве основной структуры данных использовать HashMap.

Литература:

- Дарвин, Ян. Ф. Java тм.Сборник рецептов для профессионалов / Ян. Ф. Дарвин. - СПб. [и др.] : Питер, 2002. - 764 с. : ил. - (Для профессионалов). (1)
- Эккель Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

Тема 6. Работа с файлами, потоки ввода-вывода

Содержание

- Записать случайную последовательность чисел случайной длины в файл. Прочитать файл и вывести сумму всех чисел.
- Сформировать словарь из файла с произвольным текстом.

Литература:

1. Шилдт Г. Java. Наиболее полное руководство, 8-е изд. – М.: ООО «ИД Вильямс», 2012. – 1104 с.
2. Эккель Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

Тема 7. Создание графического интерфейса программ: введение

Содержание

Написание простого интерфейса для организации диалога с пользователем. Пользователь вводит свои данные, а программа обрабатывает эти данные и выводит в метки.

Литература:

1. Шилдт Г. Java. Наиболее полное руководство, 8-е изд. – М.: ООО «ИД Вильямс», 2012. – 1104 с.

2. Эккель Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

Тема 8. Создание графического интерфейса программ: менеджеры компоновки

Содержание

Использование FlowLayout, BorderLayout, GridLayout, GridBagLayout, BoxLayout, CardLayout, AbsoluteLayout.

Литература:

1. Монахов, В.В. Язык программирования Java и среда NetBeans : [учеб. пособие] / В. В. Монахов. – 3-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2011. – 703 с. (1) (Монахов, В. Язык программирования Java и среда NetBeans. – Национальный открытый университет «Интуит». Режим доступа: <http://www.intuit.ru/studies/courses/569/425/info>)

2. Семочкин, А.Н. Язык программирования Java : учеб. пособие для студ. вузов / А. Н. Семочкин ; М-во образования и науки Рос. Федерации, Федеральное агентство по образованию [и др.]. – 2-е изд., перераб. и доп. – Благовещенск : Изд-во БГПУ, 2006, 2009. – 89 с. (12)

Тема 9. Создание графического интерфейса программ: списки, таблицы, редакторы, сложные компоненты

Содержание

Демонстрация создания электронной таблицы, списочных структур, текстовых редакторов.

Литература:

1. Шилдт Г. Java. Наиболее полное руководство, 8-е изд. – М.: ООО «ИД Вильямс», 2012. – 1104 с.

2. Эккель Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

Тема 10. Обработка строк

Содержание

Произвести разбиение строк на лексемы. Построить частотный словарь.

Литература:

1. Шилдт Г. Java. Наиболее полное руководство, 8-е изд. – М.: ООО «ИД Вильямс», 2012. – 1104 с.

2. Эккель Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

Тема 11. Сетевые средства языка java

Содержание

- Написать серверное и клиентское приложения для обмена сообщениями.
- Реализовать простейший браузер

Литература:

1. Шилдт Г. Java. Наиболее полное руководство, 8-е изд. – М.: ООО «ИД Вильямс», 2012. – 1104 с.

2. Эккель Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

6 ДИДАКТИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ КОНТРОЛЯ (САМОКОНТРОЛЯ) УСВОЕННОГО МАТЕРИАЛА

6.1 Оценочные средства, показатели и критерии оценивания компетенций

Индекс компетенции	Оценочное средство	Показатели оценивания	Критерии оценивания сформированности компетенций
ОПК-3, ПК-4, ПК-5, ПК-7	Лабораторная работа	Низкий (неудовлетворительно)	Лабораторная работа студенту не засчитывается если студент: 1. допустил число ошибок и недочетов превосходящее норму, при которой пересекается пороговый показатель; 2. или если правильно выполнил менее половины работы.
		Пороговый (удовлетворительно)	Если студент правильно выполнил не менее половины работы или допустил: 1. не более двух грубых ошибок; 2. или не более одной грубой и одной негрубой ошибки и одного недочета; 3. или не более двух-трех негрубых ошибок; 4. или одной негрубой ошибки и трех недочетов; 5. или при отсутствии ошибок, но при наличии четырех-пяти недочетов.
		Базовый (хорошо)	Если студент выполнил работу полностью, но допустил в ней: 1. не более одной негрубой ошибки и одного недочета; 2. или не более двух недочетов.
		Высокий (отлично)	Если студент: 1. выполнил работу без ошибок и недочетов; 2. допустил не более одного недочета.

6.2 Промежуточная аттестация студентов по дисциплине

Промежуточная аттестация является проверкой всех знаний, навыков и умений студентов, приобретённых в процессе изучения дисциплины. Формой промежуточной аттестации по дисциплине является экзамен.

Для оценивания результатов освоения дисциплины применяется следующие критерии оценивания.

Критерии оценивания устного ответа на экзамене

Оценка 5 (отлично) ставится, если:

- полно раскрыто содержание вопросов в объеме программы и рекомендованной литературы;

- четко и правильно даны определения и раскрыто содержание концептуальных понятий, закономерностей, корректно использованы научные термины;
- для доказательства использованы различные теоретические знания, выводы из наблюдений и опытов;
- ответ самостоятельный, исчерпывающий, без наводящих дополнительных вопросов, с опорой на знания, приобретенные в процессе специализации по выбранному направлению информатики.

Оценка 4 (хорошо) ставится, если:

- раскрыто основное содержание вопросов;
- в основном правильно даны определения понятий и использованы научные термины;
- ответ самостоятельный;
- определения понятий неполные, допущены нарушения последовательности изложения, небольшие неточности при использовании научных терминов или в выводах и обобщениях, исправляемые по дополнительным вопросам экзаменаторов.

Оценка 3 (удовлетворительно) ставится, если:

- усвоено основное содержание учебного материала, но изложено фрагментарно, не всегда последовательно;
- определение понятий недостаточно четкое;
- не использованы в качестве доказательства выводы из наблюдений и опытов или допущены ошибки при их изложении;
- допущены ошибки и неточности в использовании научной терминологии, определении понятий.

Оценка 2 (неудовлетворительно) ставится, если:

- ответ неправильный, не раскрыто основное содержание программного материала;
- не даны ответы на вспомогательные вопросы экзаменаторов;
- допущены грубые ошибки в определении понятий, при использовании терминологии.

Перечень примерных контрольных вопросов и заданий

1. Что позволяет Java-приложениям быть с легкостью перенесенными на различные платформы?
2. Почему Java устойчива к ошибкам?
3. Java интерпретируемый или компилируемый язык?
4. Как избежать путаницы, если есть два класса с одинаковыми именами?
5. Как обратиться к переменной, которая является членом класса, если внутри метода объявлена локальная с таким же именем?
6. Как вызвать конструктор родительского класса?
7. Как происходит удаление из памяти неиспользуемых объектов?
8. Может ли возникнуть ошибка нехватки памяти?
9. Как распространять программы на Java?
10. Можно ли строить в Java не прямоугольные массивы, а массивы с произвольной длиной строк?
11. Как приводить типы данных в Java?
12. Как прочитать данные из файла?
13. Как разбить строку на лексемы?
14. Как реализовать стек средствами Java?
15. Благодаря каким средствам достигается унификация внешнего вида графических приложений под различными платформами?

16. Почему не стоит всегда пользоваться абсолютным размещением графических компонентов?
17. Из какого пакета AWT или Swing следует брать управляющие компоненты для своих программ?
18. Можно ли на Java разрабатывать системные приложения?
19. Можно ли встраивать приложения Java в веб-страницы?
20. Могут ли апплеты навредить конечному пользователю и каким образом, если да?
21. По какому принципу происходит сетевое соединение клиентской и серверной части?
22. Как реализовать обработку событий на Java?
23. Можно ли создавать многопоточные приложения?
24. Существует ли в Java множественное наследование?
25. Зачем нужны интерфейсы?

Экзаменационная программа

1. История развития платформы Java, ключевые свойства технологии.
2. Принципы объектно-ориентированного программирования. Их реализация в языке Java
3. Классы, абстрактные классы, интерфейсы, пакеты в Java
4. Массивы в Java
5. Динамические структуры в Java
6. Чтение, запись, создание, удаление файлов. Изменение атрибутов файла. Просмотр содержимого директории.
7. Разработка графического интерфейса программ. Пакеты AWT, Swing.
8. Обработка исключительных ситуаций в Java.
9. Сетевые средства языка Java.
10. Сервисные классы пакета java.util.
11. Типы данных в Java.
12. Средства управления ходом выполнения программы.
13. Работа со строками в Java.

6.3 ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОВЕРКИ УРОВНЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ ОПК-3, ПК-4, ПК-5, ПК-7

Тесты содержат следующие типы заданий

Тип задания	№ задания	Вес задания (балл)	Результат оценивания (баллы, полученные за выполнение задания / характеристика правильности ответа)
задания закрытого типа с выбором одного правильного (1 из 4)	1, 2, 3	1 балл	1 б - полное правильное соответствие; 0 б - остальные случаи
задания закрытого типа с выбором одного правильного ответа по схеме: «верно»/ «неверно»	4, 5	1 балл	1 б - полное правильное соответствие; 0 б - остальные случаи

задания закрытого типа с выбором нескольких правильных ответов (3 из 6)	6, 7	2 балла	2 б – полное правильное соответствие (последовательность вариантов ответа может быть любой); 1 б – если допущена одна ошибка / ответ правильный, но не полный; 0 б – остальные случаи
задания закрытого типа на установление соответствия (4 на 4)	8, 9	2 балла	2 б – полное правильное соответствие; 1 б – если допущена одна ошибка / ответ правильный, но не полный; 0 б – остальные случаи
задание закрытого типа на установление последовательности	10, 11	2 балла	2 б – полное правильное соответствие; 1 б – если допущена одна ошибка / ответ правильный, но не полный; 0 б – остальные случаи
задания открытого типа с кратким ответом	12, 13	3 балла	3 б – полное правильное соответствие; 0 б – остальные случаи.
задания открытого типа с развернутым ответом	14, 15	5 баллов	5 б – полное правильное соответствие; если допущена одна ошибка/неточность / ответ правильный, но не полный - 3 балла; если допущено более одной ошибки / ответ неправильный / ответ отсутствует – 0 баллов

Формируемая компетенция	Индикаторы сформированности компетенции
ОПК-3. Способен понимать и применять современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения	ОПК-3.1 – знать основные положения и концепции прикладного и системного программирования, архитектуры компьютеров и сетей (в том числе и глобальных), современные языки программирования, технологии создания и эксплуатации программных продуктов и программных комплексов; ОПК-3.2 – уметь использовать их в профессиональной деятельности; ОПК-3.3 – иметь практические навыки разработки программного обеспечения

Задание 1

Внимательно прочитайте задание и укажите один правильный вариант ответа:

Какой принцип объектно-ориентированного программирования позволяет скрывать внутреннюю реализацию объекта и предоставлять только интерфейс для взаимодействия с ним?

1. наследование
2. полиморфизм
3. инкапсуляция
4. абстракция

Ответ: 3

Задание 2

Внимательно прочитайте задание и укажите один правильный вариант ответа:

Какой из перечисленных языков программирования является функциональным языком?

1. Java
2. C++
3. Python
4. Haskell

Ответ: 4

Задание 3

Внимательно прочитайте задание и укажите один правильный вариант ответа:

Какой метод в Java вызывается автоматически при создании объекта класса?

1. finalize()
2. main()
3. init()
4. constructor()

Ответ: 4

Задание 4

Верно ли следующее утверждение? В языке Java класс может наследовать функциональность сразу от нескольких родительских классов.

Ответ: неверно

Задание 5

Верно ли следующее утверждение? Динамическое связывание в объектно-ориентированном программировании позволяет определять, какой метод будет вызван, во время выполнения программы.

Ответ: верно

Задание 6

Внимательно прочитайте задание и укажите три правильных варианта ответа:

Какие из перечисленных являются принципами объектно-ориентированного программирования?

1. наследование
2. декларативность
3. инкапсуляция
4. монолитность
5. полиморфизм
6. рекурсия

Ответ: 1,3,5

Задание 7

Внимательно прочитайте задание и укажите три правильных варианта ответа:

Какие из перечисленных подходов характерны для функционального программирования?

1. иммутабельность данных
2. использование побочных эффектов
3. функции высшего порядка
4. изменение состояния
5. рекурсия вместо итераций
6. наследование классов

Ответ: 1,3,5

Задание 8

Прочитайте текст и установите соответствие:

Соотнесите парадигмы программирования с их ключевыми характеристиками:

Объектно-ориентированное программирование : Программа строится вокруг объектов, объединяющих данные и методы работы с этими данными

Функциональное программирование : Вычисления рассматриваются как вычисление математических функций без изменения состояния

Логическое программирование : Программа состоит из фактов и правил, а выполнение заключается в выводе следствий

Процедурное программирование : Программа состоит из процедур или функций, которые последовательно выполняются

Задание 9

Прочитайте текст и установите соответствие:

Соотнесите концепции ООП с их определениями:

Инкапсуляция : Механизм сокрытия внутренней реализации объекта и предоставления только интерфейса для взаимодействия

Наследование : Механизм создания нового класса на основе существующего с сохранением его свойств

Полиморфизм : Возможность объектов разных классов обрабатываться через единый интерфейс

Абстрактный класс : Класс, который не может быть инстанцирован и предназначен для наследования другими классами

Задание 10

Прочитайте текст и установите последовательность:

Расположите этапы жизненного цикла объекта в Java в правильной последовательности:

- 1 : Выделение памяти под объект
- 2 : Инициализация полей по умолчанию
- 3 : Вызов конструктора родительского класса
- 4 : Выполнение инициализаторов
- 5 : Выполнение тела конструктора
- 6 : Возврат ссылки на созданный объект

Задание 11

Прочитайте текст и установите последовательность:

Расположите в правильной последовательности шаги при работе с исключениями в Java:

- 1 : Возникновение исключительной ситуации
- 2 : Создание объекта исключения
- 3 : Поиск подходящего обработчика catch
- 4 : Передача управления обработчику
- 5 : Выполнение блока finally (если он есть)
- 6 : Продолжение выполнения программы после обработчика

Задание 12

Внимательно прочитайте задание и впишите правильный ответ: В Java механизм автоматического освобождения памяти, занятой объектами, которые больше не используются, называется _____.

Ответ: сборка мусора

Задание 13

Внимательно прочитайте задание и впишите правильный ответ: Структура данных, работающая по принципу LIFO (последним пришел - первым ушел), называется _____.

Ответ: стек

Задание 14

Внимательно прочитайте задание и запишите развернутый обоснованный ответ:

Сравните объектно-ориентированное и функциональное программирование. Какие преимущества и недостатки имеет каждый из этих подходов? В каких случаях предпочтительнее использовать каждый из них? Приведите примеры задач, которые лучше решать с использованием каждого подхода.

Ответ: Объектно-ориентированное программирование (ООП) и функциональное программирование (ФП) представляют две различные парадигмы разработки программного обеспечения.

ООП фокусируется на объектах, объединяющих данные и методы работы с этими данными. Основные принципы включают инкапсуляцию, наследование и полиморфизм. ООП хорошо подходит для моделирования реальных систем, где объекты взаимодействуют друг с другом, и для крупных проектов, где важна модульность и повторное использование кода. ФП основано на вычислении математических функций и избегает изменения состояния и изменяемых данных. Ключевые концепции включают чистые функции, функции высшего порядка, иммутабельность данных и рекурсию. ФП особенно эффективно для параллельных вычислений, обработки данных и математических расчетов.

Преимущества ООП:

- Интуитивная модель для представления реальных объектов
- Хорошая масштабируемость для больших проектов
- Легкость внесения изменений в существующий код
- Поддержка полиморфизма упрощает расширение функциональности

Недостатки ООП:

- Сложность для простых задач
- Зависимость от состояния может приводить к непредсказуемому поведению
- Сложность отладки из-за побочных эффектов
- Проблемы с параллелизмом из-за общего состояния

Преимущества ФП:

- Предсказуемость благодаря отсутствию побочных эффектов
- Простота тестирования чистых функций
- Естественная поддержка параллелизма
- Легкость рассуждения о коде

Недостатки ФП:

- Сложность понимания для начинающих
- Меньше интуитивности при моделировании реальных объектов
- Потенциальные проблемы с производительностью из-за иммутабельности
- Требуется изменения мышления от императивного подхода

ООП предпочтительно использовать для:

- Разработки GUI-приложений
- Систем моделирования реальных процессов
- Корпоративных приложений с комплексной бизнес-логикой
- Игровой разработки, где объекты взаимодействуют в сложных сценариях

ФП предпочтительно использовать для:

- Обработки и анализа данных
- Математических и научных вычислений
- Параллельных и распределенных систем

- Компиляторов и языков программирования
- Обработки событий и потоков данных (stream processing)

Задание 15

Внимательно прочитайте задание и запишите развернутый обоснованный ответ:

Объясните концепцию полиморфизма в объектно-ориентированном программировании. Какие виды полиморфизма существуют в Java? Приведите примеры использования каждого вида полиморфизма и объясните, как это повышает гибкость и расширяемость программного кода.

Ответ: Полиморфизм – один из четырех основных принципов объектно-ориентированного программирования, который позволяет объектам разных классов обрабатываться как объекты одного общего типа, при этом каждый объект может реализовывать общее поведение по-своему.

В Java выделяют несколько видов полиморфизма:

1. Параметрический полиморфизм (универсальный полиморфизм, generics):

Позволяет создавать классы, интерфейсы и методы, которые работают с различными типами данных, сохраняя при этом типобезопасность.

Преимущества: повторное использование кода, типобезопасность во время компиляции, отсутствие необходимости в приведении типов.

2. Полиморфизм подтипов (включение, наследование): Основан на возможности использовать объект дочернего класса там, где ожидается объект родительского класса. Достигается через наследование и переопределение методов.

Преимущества: возможность расширения функциональности без изменения существующего кода, реализация принципа открытости/закрытости.

3. Ад-хок полиморфизм (перегрузка методов):

Позволяет определять несколько методов с одним именем, но разными параметрами.

Преимущества: повышение читаемости кода, удобство использования API.

4. Полиморфизм через интерфейсы:

Позволяет разным классам реализовывать один интерфейс, обеспечивая единый способ взаимодействия с объектами разных типов.

Преимущества: слабая связанность компонентов, легкость замены реализаций, поддержка принципа инверсии зависимостей.

Повышение гибкости и расширяемости:

1. Легкость добавления новых типов без изменения существующего кода (принцип открытости/закрытости)
2. Возможность замены реализаций без влияния на клиентский код
3. Упрощение тестирования через использование mock-объектов
4. Улучшение читаемости и поддерживаемости кода
5. Возможность динамического выбора реализации во время выполнения

Полиморфизм позволяет создавать более абстрактные и гибкие системы, где компоненты слабо связаны между собой и могут легко заменяться или дополняться новыми реализациями, что значительно упрощает сопровождение и развитие программного обеспечения.

ПК-4. Способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях

ПК-4.1 – **знать** современные технологии проектирования и производства программного продукта;
 ПК-4.2 – **уметь** использовать подобные технологии при создании программных продуктов;
 ПК-4.3 – **иметь практический опыт** применения подобных технологий.

Задание 1

Внимательно прочитайте задание и укажите один правильный вариант ответа: Какая из перечисленных технологий Java используется для создания веб-приложений?

1. Java SE
2. Java ME
3. Java EE (Jakarta EE)
4. Java FX

Ответ: 3

Задание 2

Внимательно прочитайте задание и укажите один правильный вариант ответа: Какой компонент входит в состав JDK (Java Development Kit)?

1. только компилятор
2. только виртуальная машина Java
3. компилятор, виртуальная машина Java и библиотеки классов
4. только библиотеки классов

Ответ: 3

Задание 3

Внимательно прочитайте задание и укажите один правильный вариант ответа: Какая из перечисленных технологий является отечественной разработкой для создания программных продуктов?

1. IntelliJ IDEA
2. 1С:Предприятие
3. Eclipse
4. Visual Studio

Ответ: 2

Задание 4

Верно ли следующее утверждение? Среда разработки NetBeans является бесплатной и с открытым исходным кодом.

Ответ: верно

Задание 5

Верно ли следующее утверждение? Все современные отечественные программные продукты полностью совместимы с зарубежными аналогами.

Ответ: неверно

Задание 6

Внимательно прочитайте задание и укажите три правильных варианта ответа: Какие из перечисленных сред разработки поддерживают язык Java?

1. Eclipse
2. Visual Studio Code
3. PyCharm
4. NetBeans
5. Xcode
6. Android Studio

Ответ: 1,4,6

Задание 7

Внимательно прочитайте задание и укажите три правильных варианта ответа:

Какие из перечисленных инструментов используются для управления зависимостями в Java-проектах?

1. Maven
2. Gradle
3. npm
4. Docker
5. Jenkins
6. Ant

Ответ: 1,2,6

Задание 8

Прочитайте текст и установите соответствие:

Соотнесите компоненты Java-технологии с их назначением:

JDK (Java Development Kit) : Полный пакет для разработки Java-приложений, включающий компилятор и инструменты

JRE (Java Runtime Environment) : Среда выполнения Java-программ для конечных пользователей

JVM (Java Virtual Machine) : Виртуальная машина, обеспечивающая выполнение байт-кода Java на любой платформе

Java API : Коллекция готовых классов и интерфейсов для решения типовых задач

Задание 9

Прочитайте текст и установите соответствие:

Соотнесите виды архитектуры программного обеспечения с их характеристиками:

Клиент-серверная архитектура : Приложение разделено на клиентскую часть (представление) и серверную часть (бизнес-логика и данные)

Монолитная архитектура : Все компоненты приложения объединены в единый исполняемый файл

Микросервисная архитектура : Приложение состоит из множества небольших независимых сервисов, взаимодействующих через API

Трехуровневая архитектура : Разделение на уровень представления, бизнес-логики и данных

Задание 10

Прочитайте текст и установите последовательность:

Расположите в правильной последовательности этапы разработки программного продукта:

- 1 : Анализ требований
- 2 : Проектирование архитектуры
- 3 : Реализация (кодирование)
- 4 : Тестирование
- 5 : Разворачивание
- 6 : Сопровождение

Задание 11

Прочитайте текст и установите последовательность:

Расположите в правильной последовательности шаги компиляции Java-программы:

- 1 : Написание исходного кода с расширением .java

- 2 : Компиляция исходного кода в байт-код с помощью `javac`
- 3 : Генерация файла с расширением `.class`
- 4 : Загрузка байт-кода в JVM
- 5 : Выполнение программы JVM

Задание 12

Внимательно прочитайте задание и впишите правильный ответ: Среда выполнения Java, обеспечивающая переносимость программ между различными операционными системами, называется _____.

Ответ: JVM (Java Virtual Machine)

Задание 13

Внимательно прочитайте задание и впишите правильный ответ: Модель разработки программного обеспечения, при которой продукт создается итеративно с частыми релизами и обратной связью от пользователей, называется _____.

Ответ: Agile

Задание 14

Внимательно прочитайте задание и запишите развернутый обоснованный ответ: Опишите особенности современных отечественных информационных технологий в области разработки программного обеспечения. Какие преимущества и ограничения существуют у отечественных решений по сравнению с зарубежными аналогами? Приведите примеры успешного применения отечественных технологий в реальных проектах.

Ответ: Современные отечественные информационные технологии в области разработки ПО формируются в условиях импортозамещения, санкций и необходимости обеспечения цифрового суверенитета России. Эти технологии имеют ряд особенностей, преимуществ и ограничений.

Особенности отечественных ИТ-решений:

1. Безопасность и суверенитет:
 - Разработка с акцентом на защиту персональных данных и государственной тайны
 - Минимизация зависимости от зарубежных поставщиков
 - Поддержка отечественных стандартов шифрования и защиты информации
 - Расположение серверов и обработки данных на территории РФ
2. Государственная поддержка:
 - Финансирование через государственные программы (Цифровая экономика, ПНП "Цифровая экономика")
 - Приоритет при госзакупках программных продуктов российской разработки
 - Интеграция с государственными информационными системами (ГИС)
3. Специфика рынка:
 - Адаптация к российскому законодательству (152-ФЗ, 187-ФЗ)
 - Поддержка российской бухгалтерии и налогообложения
 - Интеграция с отечественными платежными системами (Мир, СБП)

Преимущества отечественных решений:

- Глубокая локализация и адаптация к российской специфике
- Быстрая техническая поддержка в удобное время и на русском языке
- Соблюдение требований российского законодательства
- Гибкость в модификации решений под конкретные задачи заказчика

- Меньшие затраты на лицензирование по сравнению с зарубежными аналогами
- Устойчивость к внешним санкционным ограничениям

Ограничения отечественных решений:

- Меньшее количество готовых интеграций с международными системами
- Более медленное внедрение инновационных технологий
- Ограниченный международный опыт применения
- Меньшие инвестиции в R&D по сравнению с крупными западными компаниями
- Недостаток квалифицированных специалистов для работы со специфическими решениями
- Потенциальные проблемы с масштабируемостью для очень крупных проектов

Тенденции развития:

1. Рост инвестиций в отечественные разработки
2. Повышение качества и функциональности решений
3. Формирование экосистем вокруг ключевых платформ
4. Интеграция с искусственным интеллектом и big data
5. Развитие отечественных стандартов и протоколов

В долгосрочной перспективе ожидается сближение качества отечественных и зарубежных решений при сохранении преимуществ в локализации и безопасности. Успешное развитие отечественных технологий возможно только при условии постоянного инвестирования в исследования, подготовку кадров и создание благоприятной экосистемы для разработчиков.

Задание 15

Внимательно прочитайте задание и запишите развернутый обоснованный ответ:

Объясните концепцию кроссплатформенности в современном программировании. Как Java обеспечивает кроссплатформенность приложений? Опишите преимущества и недостатки кроссплатформенных решений по сравнению с нативными приложениями для конкретных операционных систем.

Ответ: Кроссплатформенность – это способность программного обеспечения работать на различных аппаратных платформах и операционных системах без необходимости модификации или с минимальными изменениями.

Концепция кроссплатформенности основана на абстракции от деталей конкретной платформы. Вместо прямого обращения к функциям операционной системы и аппаратным ресурсам, приложение использует промежуточные слои, которые обеспечивают унифицированный интерфейс для разработчика и преобразуют вызовы в специфичные для платформы операции.

Java обеспечивает кроссплатформенность через уникальную архитектуру, построенную на трех компонентах:

1. Байт-код Java:
 - Исходный код Java компилируется не в машинный код конкретной платформы, а в специальный промежуточный формат — байт-код
 - Байт-код является платформо-независимым и одинаков для всех платформ
 - Это позволяет скомпилировать программу один раз и запустить на любом устройстве с поддержкой Java
2. Виртуальная машина Java (JVM):

- JVM – это программная среда, которая интерпретирует или компилирует байт-код в машинный код конкретной платформы
- Для каждой операционной системы и архитектуры существует своя реализация JVM
- JVM берет на себя управление памятью, безопасностью и другими низкоуровневыми аспектами

3. Java API:

- Единая библиотека классов, предоставляющая функциональность для работы с файловой системой, сетью, графикой и другими ресурсами
- Реализация этих библиотек адаптирована для каждой платформы, но интерфейс остается неизменным для разработчика

Преимущества кроссплатформенных решений:

- Экономия ресурсов на разработку: один код для нескольких платформ
- Единообразие пользовательского интерфейса на всех платформах
- Упрощение поддержки и обновления (одна версия для всех платформ)
- Сокращение времени выхода на рынок
- Широкая аудитория пользователей независимо от используемой платформы
- Упрощение командной работы (разработчики могут использовать разные ОС)

Недостатки кроссплатформенных решений:

- Снижение производительности по сравнению с нативными приложениями
- Ограниченный доступ к специфичным функциям платформы
- Проблемы с адаптацией пользовательского интерфейса под стандарты конкретной платформы
- Зависимость от поставщика кроссплатформенного фреймворка
- Дополнительные накладные расходы на абстрактные слои
- Более высокие требования к ресурсам устройства

Преимущества нативных приложений:

- Максимальная производительность и оптимизация под конкретную платформу
- Полный доступ ко всем функциям операционной системы и оборудования
- Соответствие стандартам пользовательского интерфейса платформы
- Лучшая интеграция с другими приложениями и сервисами платформы
- Независимость от сторонних фреймворков и виртуальных машин

Недостатки нативных приложений:

- Необходимость отдельной разработки для каждой платформы
- Удвоение или утроение усилий на поддержку и обновление
- Фрагментация пользовательского опыта на разных платформах
- Более высокие затраты на разработку и тестирование
- Узкая целевая аудитория

Практические подходы к обеспечению кроссплатформенности:

1. Контейнеризация (Docker):

- Упаковка приложения со всеми зависимостями в контейнер
- Обеспечивает идентичное выполнение в разных средах
- Особенно эффективна для серверных приложений

2. Веб-технологии:

- Разработка веб-приложений, работающих в браузере
- Доступны на любой платформе с поддержкой современных браузеров
- Ограничены в доступе к аппаратным ресурсам и возможностям ОС

3. Нативные обертки:

- Использование кроссплатформенных фреймворков с нативным интерфейсом
- Пример: React Native, Flutter для мобильной разработки
- Сочетает преимущества кроссплатформенности и нативного UI

4. Виртуализация и эмуляция:

- Запуск приложения в виртуальной машине или через эмулятор
- Обеспечивает максимальную совместимость, но с потерей производительности

Выбор между кроссплатформенным и нативным подходом зависит от требований проекта:

- Для корпоративных приложений часто предпочтительна кроссплатформенность
- Для игр и мультимедийных приложений важна производительность, поэтому чаще выбирают нативную разработку
- Для стартапов важна скорость выхода на рынок, поэтому часто выбирают кроссплатформенные решения
- Для приложений, интенсивно использующих специфичные функции устройства (камера, геолокация), предпочтительнее нативная разработка

Java занимает особое место в экосистеме кроссплатформенных решений благодаря своей зрелости, стабильности и широкой поддержке. Несмотря на некоторые ограничения в производительности и доступе к нативным функциям, Java остается одним из самых популярных языков для разработки кроссплатформенных приложений в корпоративном сегменте.

ПК-5. Способен использовать основные методы и средства автоматизации проектирования, реализации, испытаний и оценки качества при создании конкурентоспособного программного продукта и программных комплексов, а также способен использовать методы и средства автоматизации, связанные с сопровождением, администрированием и модернизацией программных продуктов и программных комплексов	ПК-5.1 – знать современные приемы работы с инструментальными средствами, поддерживающими создание программных продуктов и программных комплексов, их сопровождения и администрирования; ПК-5.2 – уметь использовать подобные инструментальные средства в практической деятельности; ПК-5.3 – иметь практический опыт применения подобных инструментальных средств.
---	--

Задание 1

Внимательно прочитайте задание и укажите один правильный вариант ответа: Какой из перечисленных методов сортировки имеет среднюю временную сложность $O(n \log n)$?

1. сортировка пузырьком
2. сортировка выбором
3. быстрая сортировка
4. сортировка вставками

Ответ: 3

Задание 2

Внимательно прочитайте задание и укажите один правильный вариант ответа: Какая структура данных в Java наиболее эффективна для реализации очереди?

1. ArrayList
2. LinkedList
3. HashMap
4. TreeSet

Ответ: 2

Задание 3

Внимательно прочитайте задание и укажите один правильный вариант ответа: Какой из перечисленных паттернов проектирования относится к порождающим?

1. Наблюдатель (Observer)
2. Стратегия (Strategy)
3. Одиночка (Singleton)
4. Адаптер (Adapter)

Ответ: 3

Задание 4

Верно ли следующее утверждение? Алгоритм Дейкстры может находить кратчайшие пути в графах с отрицательными весами ребер.

Ответ: неверно

Задание 5

Верно ли следующее утверждение? Бинарный поиск можно применять только к отсортированным массивам.

Ответ: верно

Задание 6

Внимательно прочитайте задание и укажите три правильных варианта ответа:

Какие из перечисленных структур данных являются линейными?

1. массив
2. связный список
3. хеш-таблица
4. дерево
5. стек
6. граф

Ответ: 1,2,5

Задание 7

Внимательно прочитайте задание и укажите три правильных варианта ответа:

Какие из перечисленных методов используются для оптимизации программ?

1. мемоизация
2. разделение и властвование
3. жадные алгоритмы
4. рекурсия
5. динамическое программирование
6. инкапсуляция

Ответ: 1,3,5

Задание 8

Прочитайте текст и установите соответствие:

Соотнесите алгоритмы сортировки с их характеристиками:

Пузырьковая сортировка : Простой, но неэффективный алгоритм с временной сложностью $O(n^2)$ в среднем случае

Быстрая сортировка : Эффективный алгоритм "разделяй и властвуй" с временной сложностью $O(n \log n)$ в среднем случае

Сортировка слиянием : Стабильный алгоритм с гарантированной временной сложностью $O(n \log n)$

Пирамидальная сортировка : Алгоритм, основанный на использовании бинарной кучи, со сложностью $O(n \log n)$

Задание 9

Прочитайте текст и установите соответствие:

Соотнесите типы тестирования ПО с их целями:

Модульное тестирование : Проверка корректности работы отдельных модулей или функций программы

Интеграционное тестирование : Проверка взаимодействия между различными компонентами системы

Системное тестирование : Проверка соответствия всей системы заданным требованиям

Регрессионное тестирование : Проверка отсутствия новых ошибок после внесения изменений в код

Задание 10

Прочитайте текст и установите последовательность: Расположите в правильной последовательности этапы решения задачи методом "разделяй и властвуй":

- 1: Разделение задачи на подзадачи
- 2: Решение каждой подзадачи независимо
- 3: Объединение решений подзадач для получения решения исходной задачи
- 4: Возврат результата

Задание 11

Прочитайте текст и установите последовательность: Расположите в правильной последовательности шаги алгоритма бинарного поиска в отсортированном массиве:

- 1: Определение середины массива
- 2: Сравнение искомого элемента с элементом в середине массива
- 3: Если элементы равны, возврат индекса середины
- 4: Если искомый элемент меньше элемента в середине, поиск продолжается в левой половине массива
- 5: Если искомый элемент больше элемента в середине, поиск продолжается в правой половине массива
- 6: Повторение процесса до нахождения элемента или определения его отсутствия

Задание 12

Внимательно прочитайте задание и впишите правильный ответ: Временная сложность операции поиска в сбалансированном бинарном дереве поиска составляет _____.

Ответ: $O(\log n)$

Задание 13

Внимательно прочитайте задание и впишите правильный ответ: Методология разработки ПО, основанная на итеративном подходе с короткими циклами разработки и частыми релизами, называется _____.

Ответ: Agile

Задание 14

Внимательно прочитайте задание и запишите развернутый обоснованный ответ: Проанализируйте выбор алгоритма для решения задачи поиска кратчайшего пути между двумя вершинами в графе. Сравните алгоритмы Дейкстры, Беллмана-Форда и A*. Какие факторы влияют на выбор конкретного алгоритма в реальных проектах? Приведите примеры предметных областей, где каждый из этих алгоритмов будет наиболее эффективен.

Ответ: Выбор алгоритма для поиска кратчайшего пути в графе зависит от множества факторов, включая характеристики графа, требования к производительности, ограничения по памяти и специфику предметной области.

Алгоритм Дейкстры: Принцип работы: Жадный алгоритм, который последовательно выбирает вершину с наименьшей текущей стоимостью пути и обновляет расстояния до соседних вершин. Характеристики:

- Временная сложность: $O((V+E)\log V)$ с использованием приоритетной очереди на основе кучи
- Работает только с неотрицательными весами ребер
- Гарантирует нахождение оптимального пути
- Требуется хранения всех вершин в памяти

Алгоритм Беллмана-Форда: Принцип работы: Итеративный алгоритм, который выполняет V-1 итерацию по всем ребрам графа для нахождения кратчайших путей. Характеристики:

- Временная сложность: $O(VE)$
- Может работать с отрицательными весами ребер
- Способен обнаруживать отрицательные циклы
- Требуется меньше памяти, чем Дейкстра
- Медленнее Дейкстры для графов без отрицательных весов

Алгоритм A*: Принцип работы: Эвристический алгоритм, расширяющий Дейкстру с использованием функции оценки для направления поиска к целевой вершине. Характеристики:

- Временная сложность: Зависит от эвристики, может быть $O(b^d)$, где b - коэффициент ветвления, d - глубина решения
- Требуется неотрицательные веса и допустимую эвристическую функцию
- Часто значительно быстрее Дейкстры при наличии хорошей эвристики
- Эффективен для поиска пути между двумя конкретными вершинами

Факторы, влияющие на выбор алгоритма:

1. Характеристики графа:
 - Размер графа (количество вершин и ребер)
 - Плотность связи между вершинами
 - Наличие отрицательных весов ребер
 - Структура графа (иерархическая, случайная, сетевая)
2. Требования к производительности:
 - Необходимость реального времени или пакетной обработки

- Частота запросов на поиск пути
- Допустимое время ответа
- 3. Ограничения по ресурсам:
 - Доступная оперативная память
 - Вычислительная мощность
 - Возможность предварительной обработки графа
- 4. Специфика задачи:
 - Необходимость поиска одного пути или всех кратчайших путей
 - Частота обновления весов ребер
 - Изменчивость структуры графа
 - Наличие дополнительных ограничений (например, ограничение на количество переходов)

Примеры предметных областей и выбор алгоритмов:

1. Системы навигации и картографические сервисы:
 - Основной алгоритм: A^* с эвристикой расстояния по прямой
 - Граф представляет дорожную сеть с расстояниями или временем в качестве весов
 - Возможны дополнительные оптимизации: иерархическая маршрутизация, contraction hierarchies
 - Требования: быстрый ответ (менее секунды), обработка миллионов запросов в день
 - Эвристика позволяет значительно сократить пространство поиска
2. Телекоммуникационные сети:
 - Основной алгоритм: Дейкстра
 - Граф представляет сеть маршрутизаторов и соединений
 - Веса ребер отражают задержки, пропускную способность или стоимость
 - Требования: надежность, гарантия оптимальности, стабильность
 - Отсутствие отрицательных весов делает Дейкстру оптимальным выбором
 - Возможно использование оптимизированных версий (bidirectional search)
3. Финансовые системы (арбитраж):
 - Основной алгоритм: Беллман-Форд
 - Граф представляет валютные пары и обменные курсы
 - Веса ребер могут быть отрицательными при представлении в логарифмической шкале
 - Задача: обнаружение отрицательных циклов для арбитража
 - Требования: точность вычислений, способность обнаруживать циклы
 - Другие алгоритмы не справляются с отрицательными весами
4. Логистика и планирование маршрутов доставки:
 - Основной алгоритм: A^* или оптимизированные версии Дейкстры
 - Граф представляет дорожную сеть с дополнительными параметрами (пробки, ограничения для грузовиков)
 - Веса могут динамически изменяться в зависимости от времени суток
 - Требования: учет множества ограничений, оптимизация по нескольким критериям

- Часто используются гибридные подходы с предварительной обработкой графа
5. Игровая индустрия (поиск пути для NPC):
 - Основной алгоритм: A* с упрощенной эвристикой
 - Граф часто представляет собой сетку или навигационную сетку уровня
 - Веса учитывают различные типы поверхностей и препятствия
 - Требования: производительность в реальном времени, предсказуемость поведения
 - Часто используются предварительно вычисленные пути и упрощенные карты
 6. Биоинформатика (анализ генетических путей):
 - Основной алгоритм: Беллман-Форд или Дейкстра
 - Графы представляют взаимодействия генов или белков
 - Веса могут быть отрицательными для обозначения ингибирующих связей
 - Требования: точность, возможность обработки больших графов
 - Беллман-Форд используется при наличии ингибирующих взаимодействий
 7. Социальные сети (анализ связей):
 - Основной алгоритм: BFS (ширина) для неориентированных графов или Дейкстра для взвешенных
 - Графы представляют связи между пользователями
 - Веса могут отражать силу связи или уровень доверия
 - Требования: масштабируемость на миллиарды вершин, распределенная обработка
 - Часто используются специализированные распределенные алгоритмы

Практические соображения в реальных проектах:

1. Гибридные подходы:
 - Комбинация нескольких алгоритмов для разных частей задачи
 - Использование мета-алгоритмов для выбора оптимального метода динамически
 - Предварительная обработка графа для ускорения поиска (contraction hierarchies, landmark-based routing)
2. Эвристические оптимизации:
 - Использование аппроксимирующих алгоритмов для получения быстрого, но не оптимального решения
 - Ограничение глубины поиска для получения приемлемого решения в ограниченное время
 - Кэширование результатов поиска для частых запросов
3. Технические аспекты:
 - Параллелизация вычислений для больших графов
 - Использование специализированного оборудования (GPU, FPGA)
 - Распределенные вычисления для обработки графов, не помещающихся в память одного сервера
4. Баланс между качеством и производительностью:
 - Определение приемлемого уровня точности для конкретной задачи

- Настройка алгоритмов для оптимизации под конкретные сценарии использования
- Мониторинг производительности и адаптация параметров в реальном времени

Выбор алгоритма всегда требует компромисса между оптимальностью результата, вычислительной сложностью, потреблением памяти и спецификой предметной области. В реальных проектах часто используются специализированные или модифицированные версии классических алгоритмов, адаптированные под конкретные требования и ограничения системы.

Задание 15

Внимательно прочитайте задание и запишите развернутый обоснованный ответ: Объясните понятие "качество программного обеспечения" и его метрики. Какие методы и инструменты используются для оценки качества ПО? Опишите современные подходы к обеспечению качества, такие как CI/CD, TDD, и их влияние на конечный продукт. Приведите примеры метрик качества для различных типов приложений.

Ответ: Качество программного обеспечения (ПО) – это степень, в которой ПО соответствует заявленным и подразумеваемым потребностям пользователей, включая функциональность, производительность, надежность, удобство использования и другие характеристики. Качество ПО является многогранным понятием, которое оценивается по множеству аспектов и метрик.

Основные аспекты качества ПО по ISO/IEC 25010:

1. Функциональная пригодность – способность ПО обеспечивать функции, удовлетворяющие заявленным и подразумеваемым потребностям
2. Производительность – производительность при заданных условиях использования
3. Совместимость – способность взаимодействовать с другими системами
4. Удобство использования — характеристики, влияющие на усилия пользователя
5. Надежность – способность выполнять требуемые функции при заданных условиях
6. Безопасность – способность защищать информацию и данные
7. Сопровождаемость – способность быть измененным для исправления дефектов или улучшения
8. Переносимость – способность быть перенесенным из одной среды в другую

Методы и инструменты оценки качества ПО:

1. Статический анализ кода:
 - Инструменты: SonarQube, Checkstyle, FindBugs, PMD
 - Метрики: сложность кода, дублирование, соответствие стандартам кодирования
 - Преимущества: выявление проблем на ранних этапах разработки
2. Тестирование:
 - Модульное тестирование (JUnit, TestNG)
 - Интеграционное тестирование
 - Системное тестирование
 - Тестирование производительности (JMeter, LoadRunner)

- Метрики: покрытие кода тестами, количество дефектов на 1000 строк кода
3. Динамический анализ:
- Профилирование производительности (VisualVM, YourKit)
 - Мониторинг памяти и использования ресурсов
 - Анализ выполнения в реальных условиях эксплуатации

4. Ручное тестирование:

- Тестирование удобства использования
- Приемочное тестирование
- Аудит безопасности

Современные подходы к обеспечению качества:

1. Непрерывная интеграция и доставка (CI/CD):

- Принцип: Автоматизация сборки, тестирования и развертывания ПО
- Инструменты: Jenkins, GitLab CI, GitHub Actions, TeamCity
- Преимущества:
 - Раннее обнаружение дефектов
 - Сокращение времени вывода функций на рынок
 - Повышение предсказуемости процесса разработки
 - Уменьшение рисков при развертывании
- Влияние на продукт: более стабильные версии, более частые обновления, меньше критических ошибок в продакшене

2. Разработка через тестирование (TDD):

- Принцип: Написание тестов перед реализацией функционала
- Цикл: красный (тест падает) -> зеленый (тест проходит) -> рефакторинг
- Преимущества:
 - Лучшая архитектура кода
 - Более полное покрытие функциональности тестами
 - Снижение количества регрессионных ошибок
 - Документация через тесты
- Влияние на продукт: более надежный код, легче поддерживаемый и расширяемый продукт

3. Мониторинг и алертинг в продакшене:

- Инструменты: Prometheus, Grafana, ELK Stack, New Relic
- Метрики: время отклика, количество ошибок, потребление ресурсов
- Преимущества:
 - Быстрое обнаружение проблем в реальном времени
 - Сбор обратной связи от реальных пользователей
 - Возможность проактивного реагирования на инциденты
- Влияние на продукт: более высокая доступность, быстрое восстановление после сбоев

4. DevOps-культура:

- Идея: Разрушение барьеров между разработкой и эксплуатацией
- Практики: Infrastructure as Code, автоматизация, совместная ответственность
- Преимущества:
 - Ускорение доставки ценности пользователям
 - Повышение качества через автоматизацию
 - Улучшение коммуникации в команде
- Влияние на продукт: более быстрая адаптация к требованиям рынка, более стабильная работа

Примеры метрик качества для различных типов приложений:

1. Веб-приложения:

- Время загрузки страницы (целевое: < 3 секунд)
- Процент успешных запросов (целевое: > 99.9%)
- Покрытие кода тестами (целевое: > 80%)
- Количество критических инцидентов в месяц (целевое: 0)
- Время восстановления после сбоя (целевое: < 15 минут)
- Уровень удовлетворенности пользователей (целевое: > 4 из 5)

2. Мобильные приложения:

- Время запуска приложения (целевое: < 2 секунды)
- Процент сбоев при запуске (целевое: < 1%)
- Рейтинг в магазине приложений (целевое: > 4.5 из 5)
- Удержание пользователей (30-дневное, целевое: > 40%)
- Размер APK/IPA файла (целевое: минимальный с сохранением функциональности)
- Потребление батареи (сравнение с конкурентами)

3. Корпоративные системы:

- Время простоя системы (целевое: < 0.1% от рабочего времени)
- Среднее время ремонта (MTTR, целевое: < 1 час)
- Количество инцидентов уровня P1 за квартал (целевое: 0)
- Скорость обработки бизнес-транзакций (целевое: в соответствии с SLA)
- Уровень соответствия требованиям безопасности (целевое: 100%)
- Количество критических уязвимостей (целевое: 0)

4. Встроенные системы:

- Надежность (MTBF - среднее время между отказами, целевое: > 10000 часов)
- Время отклика на события (целевое: в пределах спецификации)
- Потребление энергии (целевое: минимизация в рамках требований)
- Точность выполнения операций (целевое: в пределах допуска)
- Степень соответствия стандартам безопасности (целевое: 100%)
- Время восстановления после сбоя питания (целевое: минимальное)

5. Игровые приложения:

- Частота кадров (FPS, целевое: > 30 на целевых устройствах)
- Время загрузки уровня (целевое: < 30 секунд)
- Количество сбоев на 1000 игровых сессий (целевое: < 1)
- Средний сеанс игры (целевое: максимизация)

- Процент пользователей, совершающих покупки (целевое: максимизация)
- Количество отрицательных отзывов в магазине приложений (целевое: минимизация)

Комплексный подход к качеству ПО включает:

- Определение метрик качества на ранних этапах проекта
- Автоматизацию сбора и анализа метрик
- Регулярный анализ трендов и выявление проблемных областей
- Постоянное улучшение процессов разработки на основе данных
- Баланс между различными аспектами качества в соответствии с приоритетами бизнеса
- Вовлечение всех участников процесса в обеспечение качества

Качество ПО не является статичной характеристикой, а постоянно эволюционирует вместе с развитием продукта и изменениями требований пользователей. Современные подходы к обеспечению качества направлены на создание культуры непрерывного улучшения, где качество становится встроенным свойством процесса разработки, а не финальным этапом проверки перед выпуском.

ПК-7. Способен использовать основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы, способы и средства разработки программ в рамках этих направлений	ПК-7.1 – знать основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы, способы и средства разработки программ в рамках этих направлений; ПК-7.2 – уметь программировать в рамках этих направлений; ПК-7.3 – иметь практический опыт разработки программ в рамках этих направлений.
--	--

Задание 1

Внимательно прочитайте задание и укажите один правильный вариант ответа: Какой из перечисленных инструментов не является системой контроля версий?

1. Git
2. SVN (Subversion)
3. Mercurial
4. Maven

Ответ: 4

Задание 2

Внимательно прочитайте задание и укажите один правильный вариант ответа: Какой принцип SOLID гласит, что класс должен иметь только одну причину для изменения?

1. Принцип единственной ответственности (SRP)
2. Принцип открытости/закрытости (OCP)
3. Принцип подстановки Барбары Лисков (LSP)
4. Принцип разделения интерфейса (ISP)

Ответ: 1

Задание 3

Внимательно прочитайте задание и укажите один правильный вариант ответа: Какой из перечисленных типов тестирования проверяет взаимодействие между несколькими компонентами системы?

1. модульное тестирование
2. интеграционное тестирование
3. системное тестирование
4. регрессионное тестирование

Ответ: 2

Задание 4

Верно ли следующее утверждение? Контейнеризация с использованием Docker позволяет изолировать приложение и его зависимости для обеспечения единообразного выполнения в разных средах.

Ответ: верно

Задание 5

Верно ли следующее утверждение? Тестирование производительности программного обеспечения всегда проводится только после завершения функционального тестирования.

Ответ: неверно

Задание 6

Внимательно прочитайте задание и укажите три правильных варианта ответа: Какие из перечисленных практик являются частью DevOps-культуры?

1. непрерывная интеграция
2. монолитная архитектура
3. автоматизация развертывания
4. изоляция команд разработки и эксплуатации
5. инфраструктура как код
6. ручное тестирование всех изменений

Ответ: 1,3,5

Задание 7

Внимательно прочитайте задание и укажите три правильных варианта ответа: Какие из перечисленных инструментов используются для автоматизации сборки и управления зависимостями в Java-проектах?

1. Maven
2. Jenkins
3. Gradle
4. Selenium
5. JUnit
6. Ant

Ответ: 1,3,6

Задание 8

Прочитайте текст и установите соответствие: Соотнесите инструменты DevOps с их назначением:

Git : Система контроля версий для отслеживания изменений в исходном коде

Jenkins : Сервер непрерывной интеграции и доставки для автоматизации процессов сборки, тестирования и развертывания

Docker : Платформа для контейнеризации приложений с их зависимостями

Ansible : Инструмент автоматизации конфигурирования и управления инфраструктурой

Kubernetes : Система оркестрации контейнеров для управления распределенными приложениями

Prometheus : Система мониторинга и сбора метрик из приложений и инфраструктуры

Задание 9

Прочитайте текст и установите соответствие: Соотнесите типы документации ПО с их содержанием:

Техническая документация : Описание архитектуры, API, кода и внутренних процессов системы для разработчиков

Пользовательская документация : Руководства по использованию программы для конечных пользователей

Документация по установке и настройке : Инструкции по развертыванию и конфигурированию системы для администраторов

Документация по тестированию : Планы, сценарии и отчеты о тестировании для обеспечения качества

Задание 10

Прочитайте текст и установите последовательность: Расположите в правильной последовательности этапы процесса непрерывной интеграции (CI):

1 : Разработчик создает или обновляет ветку в системе контроля версий

2 : Создается запрос на слияние (pull request) в основную ветку

3 : Запускается автоматическая сборка проекта

4 : Выполняются автоматические тесты (модульные, интеграционные)

5 : Проводится статический анализ кода и проверка стиля

6 : При успешном прохождении всех шагов код сливается в основную ветку

Задание 11

Прочитайте текст и установите последовательность: Расположите в правильной последовательности шаги процесса исправления ошибки в ПО:

1 : Обнаружение и регистрация ошибки в системе отслеживания

2 : Анализ ошибки и определение ее приоритета

3 : Назначение ответственного разработчика

4 : Разработка исправления (fix)

5 : Тестирование исправления

6 : Внедрение исправления в рабочую версию и уведомление пользователя

Задание 12

Внимательно прочитайте задание и впишите правильный ответ: Процесс, при котором после успешного прохождения всех тестов изменения автоматически развертываются в production-среду, называется _____.

Ответ: непрерывное развертывание (continuous deployment)

Задание 13

Внимательно прочитайте задание и впишите правильный ответ: Методология разработки ПО, основывающаяся на итеративном подходе с короткими циклами разработки (спринтами), называется _____.

Ответ: Scrum

Задание 14

Внимательно прочитайте задание и запишите развернутый обоснованный ответ: Опишите процесс организации непрерывной интеграции и доставки (CI/CD) для Java-приложения. Какие инструменты и практики используются на каждом этапе? Какие преимущества и риски существуют при внедрении CI/CD в разработку ПО? Приведите пример конфигурации для типичного Java-проекта.

Ответ: Непрерывная интеграция и доставка (CI/CD) – это набор практик и инструментов, обеспечивающих автоматизацию процесса сборки, тестирования и развертывания программного обеспечения. Для Java-приложений этот процесс имеет свои особенности и требует использования специализированных инструментов.

Процесс организации CI/CD для Java-приложения:

1. Планирование и проектирование:
 - Определение рабочих процессов (workflows)
 - Выбор инструментов под существующую экосистему
 - Определение окружений (разработка, тестирование, stage, продакшн)
 - Проектирование стратегии ветвления (branching strategy)
 - Определение триггеров для сборки и деплоя
2. Настройка системы контроля версий:
 - Выбор системы (обычно Git)
 - Определение стратегии ветвления (Gitflow, GitHub Flow, Trunk Based Development)
 - Создание правил для pull/merge request
 - Настройка защиты основных веток
 - Установка правил именования коммитов
3. Автоматизация сборки:
 - Выбор системы сборки (Maven, Gradle или Ant)
 - Настройка зависимостей
 - Определение профилей сборки для разных окружений
 - Настройка генерации артефактов (JAR, WAR, EAR)
 - Оптимизация времени сборки
4. Автоматизация тестирования:
 - Настройка фреймворков для модульных тестов (JUnit, TestNG)
 - Интеграция с mocking-библиотеками (Mockito, PowerMock)
 - Настройка интеграционных тестов (TestContainers для баз данных)
 - Интеграция с фреймворками для UI-тестирования (Selenium)
 - Анализ покрытия кода тестами (JaCoCo)
 - Определение пороговых значений для качества
5. Статический анализ кода:
 - Интеграция с инструментами анализа (SonarQube, Checkstyle, PMD)
 - Определение правил и стандартов кодирования
 - Настройка качества порогов
 - Интеграция с pull request для анализа изменений
6. Развертывание:

- Настройка сред выполнения (Tomcat, WildFly, Spring Boot)
- Создание конфигураций для разных окружений
- Использование контейнеризации (Docker)
- Настройка оркестрации (Kubernetes)
- Реализация стратегий развертывания (blue/green, canary)

7. Мониторинг и обратная связь:

- Интеграция с системами мониторинга (Prometheus, Grafana)
- Настройка сбора логов (ELK Stack, Graylog)
- Интеграция с системами оповещения
- Визуализация метрик и состояния pipeline
- Анализ производительности и стабильности

Инструменты для CI/CD в Java-разработке:

1. Системы CI/CD:

- Jenkins — наиболее распространенный open-source сервер
- GitLab CI/CD — тесная интеграция с GitLab
- GitHub Actions — для проектов на GitHub
- TeamCity — коммерческое решение от JetBrains
- Bamboo — коммерческое решение от Atlassian

2. Сборка и управление зависимостями:

- Maven — стандарт де-факто для Java-проектов
- Gradle — более гибкий и быстрый альтернативный вариант
- Ant — устаревающий, но все еще используемый инструмент

3. Контейнеризация и оркестрация:

- Docker — для создания и управления контейнерами
- Kubernetes — для оркестрации контейнеров
- Docker Compose — для локальной разработки и тестирования

4. Мониторинг и логирование:

- Prometheus + Grafana — для сбора и визуализации метрик
- ELK Stack (Elasticsearch, Logstash, Kibana) — для централизованного логирования
- Jaeger/Zipkin — для распределенной трассировки

5. Тестирование:

- JUnit/TestNG — для модульных тестов
- Mockito/PowerMock — для мокирования зависимостей
- Selenium — для автоматизации UI-тестов
- JMeter/Gatling — для нагрузочного тестирования

Преимущества внедрения CI/CD:

1. Повышение качества кода:

- Раннее обнаружение ошибок и конфликтов
- Автоматизированное тестирование на всех уровнях
- Стандартизация кода через статический анализ
- Постоянное улучшение технического долга

2. Ускорение доставки ценности:

- Сокращение времени от идеи до реализации
- Более частые и предсказуемые релизы
- Быстрое получение обратной связи от пользователей
- Способность быстро реагировать на изменения рынка

3. Снижение рисков:

- Меньший размер изменений снижает риск ошибок
- Автоматизация уменьшает человеческий фактор
- Возможность быстрого отката при проблемах
- Поэтапное развертывание снижает влияние сбоев

4. Повышение продуктивности команды:

- Снижение рутины и повторяющихся задач
- Освобождение времени для решения сложных задач
- Улучшение коммуникации и прозрачности
- Стандартизация процессов разработки

5. Улучшение надежности:

- Предсказуемость процесса развертывания
- Повторяемость операций
- Снижение количества инцидентов в продакшене
- Быстрое восстановление после сбоев

Риски внедрения CI/CD:

1. Первоначальные затраты:

- Значительные временные затраты на настройку
- Необходимость обучения команды новым практикам
- Переработка существующих процессов разработки
- Финансовые затраты на инфраструктуру и инструменты

2. Сложность и поддержка:

- Увеличение сложности инфраструктуры
- Необходимость постоянного обновления и поддержки pipeline
- Зависимость от множества инструментов и интеграций
- Сложность отладки проблем в автоматизированных процессах

3. Культурные изменения:

- Сопротивление команды изменениям в процессах
- Необходимость изменения менталитета разработчиков
- Баланс между скоростью и качеством
- Разделение ответственности между командами

4. Потенциальные проблемы:

- Перегрузка системы сборки при большом количестве разработчиков
- Ложные срабатывания тестов или анализаторов
- Зависимость от стабильности внешних сервисов
- Проблемы с безопасностью автоматизированных процессов

Ключевые аспекты эффективного CI/CD для Java:

1. Оптимизация производительности:

- Использование кэширования зависимостей
 - Параллельное выполнение независимых задач
 - Инкрементальная сборка вместо полной
 - Выбор правильных образов контейнеров
2. Безопасность:
- Хранение секретов в защищенных переменных
 - Регулярное обновление зависимостей
 - Сканирование образов на уязвимости
 - Ограничение прав доступа к CI/CD системе
3. Масштабируемость:
- Распределение нагрузки между агентами сборки
 - Использование облачных ресурсов по требованию
 - Оптимизация использования памяти и CPU
 - Горизонтальное масштабирование инфраструктуры
4. Надежность:
- Настройка автоматических повторных попыток
 - Мониторинг состояния pipeline
 - Регулярное резервное копирование конфигураций
 - Документирование процессов восстановления
5. Гибкость:
- Параметризация конфигураций
 - Поддержка нескольких окружений
 - Возможность перезапуска отдельных этапов
 - Адаптация под разные типы проектов

Внедрение CI/CD для Java-приложений требует системного подхода, учитывающего как технические аспекты, так и организационные изменения в команде. Успешная реализация приводит к значительному повышению качества и скорости доставки программного обеспечения, что в конечном итоге обеспечивает конкурентное преимущество на рынке.

Задание 15

Объясните роль паттернов проектирования в разработке программного обеспечения. Какие основные категории паттернов существуют? Приведите примеры паттернов из каждой категории и опишите, как они помогают решать типовые проблемы при проектировании ПО. Объясните, как использование паттернов влияет на поддерживаемость, расширяемость и качество кода. Когда не следует использовать паттерны проектирования?

Ответ: Паттерны проектирования – это проверенные решения типовых проблем, возникающих при разработке программного обеспечения. Они представляют собой шаблоны для решения задач, которые можно применять в различных контекстах, позволяя разработчикам избегать изобретения велосипеда и использовать проверенные практики.

Роль паттернов проектирования в разработке ПО:

1. Улучшение качества архитектуры:
 - Предоставляют проверенные решения для типовых проблем
 - Помогают создавать гибкие и расширяемые архитектуры
 - Снижают связанность компонентов системы

- Улучшают повторное использование кода
- 2. Упрощение коммуникации в команде:
 - Создают общий язык для разработчиков
 - Позволяют быстро передавать сложные идеи через названия паттернов
 - Упрощают документирование архитектурных решений
 - Снижают кривую обучения для новых членов команды
- 3. Повышение эффективности разработки:
 - Сокращают время на проектирование
 - Уменьшают количество архитектурных ошибок
 - Ускоряют рефакторинг кода
 - Помогают избежать распространенных ловушек проектирования

Основные категории паттернов проектирования по классификации "Банды четырех" (GoF):

1. Порождающие паттерны (Creational Patterns): Управляют процессом создания объектов, делая систему независимой от того, как, из чего и когда эти объекты создаются.
2. Структурные паттерны (Structural Patterns): Заботятся о компоновке классов и объектов для формирования больших структур.
3. Поведенческие паттерны (Behavioral Patterns): Отвечают за эффективное взаимодействие между объектами и распределение обязанностей.

Когда не следует использовать паттерны проектирования:

1. Перегрузка простых решений
2. Преждевременная оптимизация
3. Непонимание контекста
4. Сложность для команды
5. Производительность

Паттерны проектирования являются мощным инструментом в арсенале разработчика, но как любой инструмент, они требуют правильного применения. Опытные разработчики умеют распознавать ситуации, когда паттерн действительно необходим, и когда его применение приведет к избыточной сложности. Важно помнить, что цель паттернов – сделать код более понятным, поддерживаемым и гибким, а не просто следовать моде или "лучшим практикам" без понимания их смысла.

7 ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ В ПРОЦЕССЕ ОБУЧЕНИЯ

Информационные технологии – обучение в электронной образовательной среде с целью расширения доступа к образовательным ресурсам, увеличения контактного взаимодействия с преподавателем, построения индивидуальных траекторий подготовки, объективного контроля и мониторинга знаний студентов.

В образовательном процессе по дисциплине используются следующие информационные технологии, являющиеся компонентами Электронной информационно-образовательной среды БГПУ:

- официальный сайт БГПУ;
- корпоративная сеть БГПУ;
- система электронного обучения ФГБОУ ВО «БГПУ»;
- электронные библиотечные системы;

- средства разработки Java: JDK, JRE, NetBeans.

8 ОСОБЕННОСТИ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ ИНВАЛИДАМИ И ЛИЦАМИ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

При обучении лиц с ограниченными возможностями здоровья применяются адаптивные образовательные технологии в соответствии с условиями, изложенными в раздел «Особенности организации образовательного процесса по образовательным программам для инвалидов и лиц с ограниченными возможностями здоровья» основной образовательной программы (использование специальных учебных пособий и дидактических материалов, специальных технических средств обучения коллективного и индивидуального пользования, предоставление услуг ассистента (помощника), оказывающего обучающимся необходимую техническую помощь и т.п.) с учётом индивидуальных особенностей обучающихся.

9 СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ

9.1 Литература

1. Блох Д. Java. Эффективное программирование. 2-е изд. – М.: Издательство Лори, 2014. – 461 с. (7 экз.)
2. Зыков, С. В. Программирование. Объектно-ориентированный подход : учебник и практикум для вузов / С. В. Зыков. – Москва : Издательство Юрайт, 2022. – 155 с. – (Высшее образование). – ISBN 978-5-534-00850-0. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: <https://urait.ru/bcode/490423> (дата обращения: 20.10.2022).
3. Монахов, В.В. Язык программирования Java и среда NetBeans : [учеб. пособие] / В. В. Монахов. – 3-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2011. – 703 с. (Монахов, В. Язык программирования Java и среда NetBeans. – Национальный открытый университет «Интуит». – Режим доступа : <https://intuit.ru/studies/courses/569/425/info>)
4. Ноутон, П. Java 2 : перевод с английского / П. Ноутон, Г. Шилдт. – Дюссельдорф; Киев ; М. ; СПб. : БхВ - Петербург, 2001. – 1050 с. (10 экз.)
5. Стелтинг, С. Применение шаблонов Java тм : [Справоч. руководство разработчика по архитектуре шаблонов для платформы Java] / С. Стелтинг, О. Маассен. – М. ; СПб. ; Киев : Вильямс, 2002. – 563 с. – (Библиотека профессионала). (6 экз.)

9.2 Базы данных и информационно-справочные системы

1. Федеральный портал «Российское образование» – Режим доступа: <http://www.edu.ru>
2. Информационная система «Единое окно доступа к образовательным ресурсам» – Режим доступа: <http://www.window.edu.ru>
3. Федеральный центр информационно-образовательных ресурсов – Режим доступа: <http://fcior.edu.ru>
4. Сайт Федеральной службы по интеллектуальной собственности, патентам и товарным знакам (Роспатента). – Режим доступа: <http://www.fips.ru/rospatent/index.htm>

9.3 Электронно-библиотечные ресурсы

1. ЭБС «Юрайт». – Режим доступа: <https://urait.ru>
2. Полпред (обзор СМИ). – Режим доступа: <https://polpred.com/news>

10 МАТЕРИАЛЬНО-ТЕХНИЧЕСКАЯ БАЗА

Для проведения занятий лекционного и семинарского типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации используются аудитории, оснащённые учебной мебелью, аудиторной доской, компьютером(рами) с установленным лицензионным специализированным программным обеспечением, коммутатором для выхода в электронно-библиотечную систему и электронную информационно-образовательную среду БГПУ, мультимедийными проекторами, экспозиционными экранами, учебно-наглядными пособиями мультимедийные презентации).

Самостоятельная работа студентов организуется в аудиториях оснащенных компьютерной техникой с выходом в электронную информационно-образовательную среду вуза, в специализированных лабораториях по дисциплине, а также в залах доступа в локальную сеть БГПУ, в лаборатории психолого-педагогических исследований и др.

Лицензионное программное обеспечение: операционные системы семейства Windows, Linux; офисные программы Microsoft office, Libreoffice, OpenOffice; средства разработки Java: JDK, JRE, NetBeans.

Разработчик: Ягелло А.А., ст. преп. кафедры информатики и МПИ.

11 ЛИСТ ИЗМЕНЕНИЙ И ДОПОЛНЕНИЙ

Утверждение изменений и дополнений в РПД для реализации в 2024/2025 уч. г.

РПД обсуждена и одобрена для реализации в 2024/2025 уч. г. на заседании кафедры информатики и МПИ (протокол №8 от 29 мая 2024 г.).

Утверждение изменений и дополнений в РПД для реализации в 2025/2026 уч. г.

РПД обсуждена и одобрена для реализации в 2025/2026 уч. г. на заседании кафедры информатики и МПИ (протокол №6 от 26 марта 2025 г.).

Утверждение изменений и дополнений в РПД для реализации в 2026/2027 уч. г.

РПД обсуждена и одобрена для реализации в 2026/2027 уч. г. на заседании кафедры информатики и МПИ (протокол №7 от 26 мая 2026 г.).