

Федеральное государственное образовательное бюджетное учреждение высшего образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)

Кафедра информационных технологий
Факультет информационных технологий и анализа больших данных

Документ подписан усиленной неквалифицированной электронной подписью.

Организация: «ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ
РОССИЙСКОЙ ФЕДЕРАЦИИ»

Сертификат: WEtREGD7OIda6eA+7USk0Jm3If1URR7

Утверждено: Проректор по учебной и методической работе, Е.А. Каменева

Дата: 11.02.2026 г.

С.А. Румянцев

Объектно-ориентированное проектирование

Рабочая программа дисциплины

для студентов, обучающихся по направлению подготовки

09.03.03 - Прикладная информатика,

Образовательная программа «Прикладные информационные системы в экономике и
финансах»

Профиль:

«Прикладные информационные системы в экономике и финансах»

Рекомендовано

*Факультет информационных технологий и анализа больших данных
(протокол № 07 от 22.04.2026 г.)*

Одобрено

*Кафедра информационных технологий
(протокол № 04 от 13.04.2026 г.)*



Содержание

1. Наименование дисциплины	3
2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине	3
3. Место дисциплины в структуре образовательной программы	
4. Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся	5
5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий	6
5.1. Содержание дисциплины	6
5.2. Учебно – тематический план	10
5.3. Содержание семинаров, практических занятий	12
6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине	18
6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	18
6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю	30
7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине	31
8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	7
9. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	8
10. Методические указания для обучающихся по освоению дисциплины	40
11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)	42
12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	42



1. Наименование дисциплины

Объектно-ориентированное проектирование

2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине

Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции
ПКП-5	Способен разрабатывать требования и проектировать программное обеспечение	Анализирует возможности реализации требований к компьютерному программному обеспечению	Знать: методы анализа и формализации требований к ПО; основные подходы к проектированию программного обеспечения; Уметь: анализировать и формализовывать требования; оценивать их реализуемость;
		Проектирует компьютерное программное обеспечение	Знать: архитектурные стили и шаблоны проектирования; принципы ООП и декомпозиции систем; Уметь: проектировать архитектуру и структуру ПО; разрабатывать модели системы и обосновывать решения;
ПКП-8	Способен выполнять работы и управлять работами по созданию (модификации) и сопровождению информационных систем (ИС), автоматизирующих задачи организационного управления и бизнес-процессы	Разрабатывает модели бизнес-процессов заказчика в рамках проекта создания (модификации) ИС	Знать: методы моделирования бизнес-процессов (BPMN, UML); основы анализа предметной области и требований заказчика; Уметь: моделировать бизнес-процессы заказчика; анализировать требования и формализовывать их;
		Разрабатывает архитектуру ИС в рамках выполнения работ и управ-	Знать: принципы построения архитектуры информационных систем; типовые архитектурные решения ИС;



Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции
		ления работами по созданию (модификации) и сопровождению ИС	Уметь: проектировать архитектуру информационной системы; выбирать и обосновывать архитектурные решения;



3. Место дисциплины в структуре образовательной программы

Дисциплина «Объектно-ориентированное проектирование» относится к «Модулю "Прикладное программирование"».

4. Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся

Очная форма обучения

Вид учебной работы по дисциплине	Всего (в з/е и часах)	Семестр 6 (в часах)
Общая трудоемкость дисциплины	3/108	108
Контактная работа – Аудиторные занятия	34	34
<i>Лекции</i>	<i>16</i>	<i>16</i>
<i>Семинары, практические занятия</i>	<i>18</i>	<i>18</i>
<i>Самостоятельная работа</i>	<i>74</i>	<i>74</i>
Вид текущего контроля	Контрольная работа	Контрольная работа
Вид промежуточной аттестации	Зачет	Зачет



5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий

5.1. Содержание дисциплины

Тема 1. Введение в объектно-ориентированное проектирование и Java

Парадигмы программирования: процедурная vs объектно-ориентированная. Основные принципы ООП: абстракция, инкапсуляция, наследование, полиморфизм. Обзор платформы Java: JVM, байт-код, JDK. Структура простой Java-программы. Синтаксис языка Java: типы данных, переменные, операторы, управляющие конструкции. Среда разработки (IDE) и инструменты сборки.

Тема 2. Классы и объекты в Java. Инкапсуляция

Определение классов, полей и методов. Создание объектов (оператор new). Конструкторы и перегрузка методов. Модификаторы доступа (public, private, protected) и организацию encapsulation. Пакеты и организационное разделение кода на модули. Примеры реализации классов для экономических задач (например, класс BankAccount). Практика проектирования классов: принцип единственной ответственности.

Тема 3. Наследование и полиморфизм

Иерархия классов, ключевое слово extends. Базовый и производный классы, переопределение методов (@Override). Виртуальные методы и позднее связывание (полиморфизм во время выполнения). Класс Object и методы toString(), equals() и др. Абстрактные классы и методы, интерфейсы (ключевое слово implements): различия и использование. Пример: модель «Сотрудник–Менеджер–Директор» с общим базовым классом. Интерфейсы Java API (Comparable, Serializable и др.). Использование полиморфизма для расширяемости программ.

Тема 4. Обработка исключений

Механизм исключений в Java: try-catch-finally, множество catch и иерархия исключений (Exception, RuntimeException, Error). Проверяемые и непроверяемые исключения. Оператор throws и проброс исключений. Создание собственных классов исключений (напр., InsufficientFundsException). Логирование и диагностика ошибок. Практические примеры: обработка ошибок ввода пользователя, работа с



исключениями при вычислениях в финансах (деление на ноль, неверный формат числа).

Тема 5. Коллекции данных и обобщения (Generics)

Коллекции в Java (Java Collections Framework): списки (List, реализация ArrayList/LinkedList), множества (Set, HashSet/TreeSet), очереди, словари (Map, HashMap/TreeMap). Основные операции над коллекциями: добавление, удаление, поиск, сортировка (Comparator, Comparable). Итераторы. Обобщения (generic types): параметризация классов и методов типами, <> синтаксис. Ограничения и особенности дженериков (type erasure). Пример: хранение и обработка списка финансовых транзакций. Эффективность различных коллекций.

Тема 6. Ввод-вывод в Java. Потоки ввода-вывода

Байтовые и символьные потоки, классы InputStream/OutputStream и Reader/Writer. Чтение и запись текстовых файлов (FileReader/Writer, BufferedReader/Writer). Работа с файловой системой (класс File, Paths, Files). Обработка исключений при вводе-выводе (IOException). Кодировки символов. Пример: чтение CSV-файла с данными и формирование отчета. Кратко о NIO (New I/O) и Channels/Buffers.

Тема 7. Функциональное программирование в Java: лямбда-выражения и Stream API

Понятие лямбда-выражения, синтаксис и области применения. Функциональные интерфейсы (Predicate, Function, Consumer и др.) и аннотация @FunctionalInterface. Использование лямбд для сортировки, фильтрации (Comparator на базе лямбда). Потоки (streams) в Java 8+: создание потока из коллекции, промежуточные операции (filter, map, sorted) и терминальные операции (forEach, collect, reduce). Понятие неизменяемости и отложенного выполнения в Stream API. Пример: фильтрация списка транзакций по критерию и агрегирование результатов.

Тема 8. Аннотации и рефлексия

Аннотации в Java: встроенные (@Override, @Deprecated, @SuppressWarnings) и кастомные аннотации. Объявление собственной аннотации (@interface), элементы аннотации. Использование аннотаций в фреймворках (например, @Test в JUnit, аннотации в Spring для конфигурации бинов). Рефлексия: класс Class, получение информации о типе во время выполнения, использование java.lang.reflect (Method, Field, Constructor) для динамического вызова методов и создания объектов. Безопасность и производительность рефлексии.



Тема 9. Модульная система Java

Концепция модулей (Java Platform Module System, JPMS) введена в Java 9: описание модуля (файл module-info.java), экспорт пакетов, требование модулей. Организация крупного приложения в виде модулей, инкапсуляция через модули. Преимущества модульности (лучшее управление зависимостями, уменьшение размера JRE через jlink). Пример: создание простого модуля и подключение библиотеки как модуля.

Тема 10. Современные технологии корпоративной разработки на Java: Java EE и Spring

Обзор платформы Java EE (Jakarta EE): компоненты корпоративного приложения (серветы, JSP, EJB, JDBC, JMS и др.), трёхуровневая архитектура (клиент – сервер приложений – СУБД). Ограничения классической Java EE (сложность конфигурации, громоздкость приложений). Введение в Spring Framework – легковесный контейнер для создания enterprise-приложений. Принципы IoC (Inversion of Control) и DI (Dependency Injection) в Spring Spring Boot как современный подход к быстрой разработке: авто-конфигурация, встроенный веб-сервер. Создание простейшего веб-приложения REST на Spring Boot (контроллер, сервис, репозиторий – в обзоре). Особенности Spring: аннотации (@Component, @Autowired, @Controller, @Service и т.д.), управление транзакциями, интеграция с БД (Spring Data). Значение Spring Framework в индустрии (де-факто стандарт enterprise-разработки на Java).

Тема 11. Шаблоны проектирования в объектно-ориентированных системах

Понятие паттерна проектирования; обзор основных категорий GOF-паттернов: порождающие, структурные, поведенческие. Разбор примеров популярных шаблонов: Singleton (единственный экземпляр), Factory Method и Abstract Factory (создание объектов), Observer (наблюдатель), Strategy (стратегия), Decorator (декоратор) и др. Применение шаблонов на языке Java: реализация и типичные случаи использования. Влияние паттернов на архитектуру и поддерживаемость кода. Практический пример: использование паттерна Наблюдатель для оповещения об изменении финансовых показателей.

Тема 12. Сравнение ООП-языков: Java, C++, C#, Python

Обзор реализации ООП-концепций в разных языках: классы и прототипы, статическая vs динамическая типизация, сборка мусора vs ручное управление памятью (C++), поддержка множественного наследования (в C++ и Python) vs интерфейсов (Java, C#). Управление памятью: сборщик мусора в Java/Python vs delete



в C++. Особенности C#: сходство с Java (CLR vs JVM, LINQ, делегаты). Python: динамическое ООП, duck typing. Краткий анализ – сильные и слабые стороны Java по сравнению с другими (безопасность, производительность, простота разработки).

Тема 13. Юнит-тестирование и обеспечение качества кода

Подходы к тестированию программного обеспечения: модульное (unit testing), интеграционное, системное тестирование. JUnit – фреймворк модульного тестирования для Java: написание тестовых методов с аннотациями @Test, Assertions для проверки результатов. Организация набора тестов, отчет о покрытии. Принципы TDD (разработка через тестирование). Использование логгеров (SLF4J, Log4j) для отслеживания выполнения. Инструменты анализа кода (Checkstyle, SonarQube – обзор). Практика: написание тестов для ранее реализованных классов (например, тестирование методов расчета финансовой формулы).



5.2. Учебно – тематический план

Очная форма обучения

п/ п	Наименование тем (разделов) дисциплины	Трудоемкость в часах				
		Всего	Контактная работа - Аудиторная работа			Самостоятельная работа
			Общая, в т.ч.:	Лекции	Семинары, практические занятия	
1	Введение в объектно-ориентированное проектирование и Java	9	4	2	2	5
2	Классы и объекты в Java. Инкапсуляция	8	3	1	2	5
3	Наследование и полиморфизм	8	3	1	2	5
4	Обработка исключений	8	3	1	2	5
5	Коллекции данных и обобщения (Generics)	9	2	1	1	7
6	Ввод-вывод в Java. Потоки ввода-вывода	9	3	1	2	6
7	Функциональное программирование в Java: лямбда-выражения и Stream API	9	3	1	2	6
8	Аннотации и рефлексия	8	2	1	1	6
9	Модульная система Java	7	1	1	0	6
10	Современные технологии корпоративной	11	3	2	1	8



п/ п	Наименование тем (разделов) дисциплины	Трудоемкость в часах				
		Всего	Контактная работа - Аудиторная работа			Самостоятельная работа
			Общая, в т.ч.:	Лекции	Семинары, практические занятия	
	разработки на Java: Java EE и Spring					
11	Шаблоны проектирования в объектно-ориентированных системах	10	3	2	1	7
12	Сравнение ООП-языков: Java, C++, C#, Python	6	2	1	1	4
13	Юнит-тестирование и обеспечение качества кода	6	2	1	1	4
В целом по дисциплине		108	34	16	18	74



5.3. Содержание семинаров, практических занятий

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Введение в объектно-ориентированное проектирование и Java	• В чем ключевые различия процедурного и объектно-ориентированного подходов? • Что означают принципы ООП: абстракция, инкапсуляция, наследование, полиморфизм? • Что такое JVM, JDK, JRE и как они связаны между собой? • Что такое байт-код и как выполняется Java-приложение? • В чем смысл принципа «Write Once, Run Anywhere» и за счет чего он реализуется?	Практические (семинарские) занятия
Классы и объекты в Java. Инкапсуляция	• Чем отличаются класс и объект в Java? • Что такое поля, методы, конструкторы; зачем нужна перегрузка? • Как работают модификаторы доступа и какие задачи решает инкапсуляция? • Зачем нужны пакеты и как они помогают организации кода? • В чем смысл принципа единственной ответственности (SRP) при проектировании классов?	Практические (семинарские) занятия



Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Наследование и полиморфизм	• Что такое наследование и как используется extends? • Чем отличается перегрузка от переопределения методов? • Что такое полиморфизм времени выполнения и позднее связывание? • Абстрактные классы и интерфейсы: в чем различия и когда что применять? • Какую роль играет класс Object и методы toString(), equals(), hashCode()?	Практические (семинарские) занятия
Обработка исключений	• Что такое исключения и как устроена их иерархия в Java? • Checked vs unchecked: в чем разница и как это влияет на проектирование? • Как работают try-catch-finally, throw и throws? • Зачем создавать пользовательские исключения и какие правила для них важны? • Какую роль играют логирование и диагностика ошибок?	Практические (семинарские) занятия
Коллекции данных и обобщения (Generics)	• Чем отличаются List, Set, Map и когда применять каждую структуру? • ArrayList vs LinkedList: отличия по сложности операций и сценарии выбора.	Практические (семинарские) занятия



Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
	• HashMap vs TreeMap, HashSet vs TreeSet: чем отличаются и что дают сортируемые структуры? • Зачем нужны Comparable и Comparator? • Для чего нужны generics и какие у них ограничения (type erasure)?	
Ввод-вывод в Java. Потоки ввода-вывода	• Байтовые и символьные потоки: в чем различия и что выбрать? • Зачем используются буферизированные потоки (Buffered*)? • Как работают File/Paths/Files и чем подход NIO отличается от классического I/O? • Что такое кодировка и какие проблемы она вызывает при чтении/записи текстов? • Как корректно обрабатывать IOException?	Практические (семинарские) занятия
Функциональное программирование в Java: лямбда-выражения и Stream API	• Что такое лямбда-выражение и функциональный интерфейс? • Какие задачи решают Predicate, Function, Consumer и т.п.? • Как устроен конвейер Stream: промежуточные и терминальные операции? • Что означает «ленивость» вычислений в Stream API? • В чем особенности и риски parallelStream()?	Практические (семинарские) занятия



Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Аннотации и рефлексия	• Что такое аннотации и какие бывают Retention/Target? • Как создать собственную аннотацию и где она используется? • Что такое reflection и какие базовые операции она позволяет выполнять? • Какие типовые сценарии применения рефлексии в Java-фреймворках? • Каковы ограничения/риски рефлексии (производительность, безопасность, инкапсуляция)?	Практические (семинарские) занятия
Модульная система Java	-	-



Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Современные технологии корпоративной разработки на Java: Java EE и Spring	• Чем концептуально отличаются подходы Java EE (Jakarta EE) и Spring? • Что такое IoC и DI и почему они важны в корпоративной разработке? • Какую роль играют основные стереотипы/аннотации Spring (@Component, @Service, @Controller, @Autowired)? • Зачем нужен Spring Boot и что дает автоконфигурация? • Как выглядит типовая слоистая архитектура приложения (controller/service/repository)?	Практические (семинарские) занятия
Шаблоны проектирования в объектно-ориентированных системах	• Что такое паттерн проектирования и чем он отличается от «архитектурного стиля»? • Какие есть группы паттернов (порождающие/структурные/поведенческие)? • В каких ситуациях уместны Singleton и Factory Method? • В чем идея Observer и Strategy (задачи, которые они решают)? • Как связаны принципы SOLID и применение паттернов?	Практические (семинарские) занятия
Сравнение ООП-языков: Java, C++, C#, Python	• Чем отличаются модели исполнения и платформы JVM/.NET/нативная компиляция?	Практические (семинарские) занятия



Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
	• Статическая и динамическая типизация: плюсы/минусы на практике. • Как различается управление памятью (GC vs manual) и последствия для разработки? • Как в языках реализуются наследование/интерфейсы/множественное наследование? • Какие типовые области применения сильнее у Java по сравнению с C++/C#/Python?	
Юнит-тестирование и обеспечение качества кода	• Что такое модульное тестирование и какую проблему оно решает? • Базовая структура теста в JUnit и роль assertions. • Что такое TDD и когда оно оправдано? • Что означает «покрытие кода тестами» и как его правильно интерпретировать? • Какие практики повышают тестопригодность кода?	Практические (семинарские) занятия



6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине

6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы

Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
Введение в объектно-ориентированное проектирование и Java	В чем различие процедурного и объектно-ориентированного подходов (по целям, структуре кода и сопровождению)? Что означает каждый из принципов ООП: инкапсуляция, наследование, полиморфизм, абстракция? Что такое JVM, JDK и JRE: назначение каждого компонента. Что такое байт-код и каков общий цикл компиляции/запуска Java-программы? Что означает принцип «Write Once, Run Anywhere» и какие ограничения у него есть? Какова роль IDE и инструментов сборки (Maven/Gradle) в жизненном цикле Java-проекта?	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.
Классы и объекты в Java. Инкапсуляция	Что такое класс и объект в языке Java. Как они соотносятся с понятиями в ООП. Какие члены класса (поля, методы, блоки инициализации, внутренние классы) существуют в Java. Различия между методами экземпляра и статическими методами. Понятие инкапсуляции. Зачем она нужна при проектировании программных систем.	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
	Что такое модификаторы доступа (public, private, protected, default) и как они ограничивают доступ к членам класса. Как реализуется принцип единственной ответственности в проектировании классов. Назначение геттеров и сеттеров. Почему прямой доступ к полям нарушает инкапсуляцию. Автоматическая генерация методов toString(), equals() и hashCode() в IDE. Что означает перегрузка методов (method overloading) и как она реализуется в Java. Использование классов-оболочек (Integer, Double и др.) и их взаимодействие с объектами и примитивами. Отличие между this и ссылкой на объект. Случаи, где this необходим.	
Наследование и полиморфизм	Понятие и назначение наследования в объектно-ориентированном программировании. Ключевое слово extends. Различие между абстрактным классом и интерфейсом. Когда использовать каждый из них. Механизм переопределения методов. Использование аннотации @Override. Понятие полиморфизма. Виды полиморфизма в Java. Позднее связывание (dynamic dispatch) и его влияние на архитектуру приложений.	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
	Использование ключевых слов <code>super</code> и <code>this</code> в контексте иерархии классов. Ограничения множественного наследования в Java и способы их обхода с помощью интерфейсов. Иерархия классов в стандартной библиотеке Java (<code>Object</code>, <code>Comparable</code>, <code>Serializable</code> и др.).	
Обработка исключений	Что такое исключительная ситуация (exception) в контексте исполнения программы. Иерархия классов исключений в Java: <code>Throwable</code>, <code>Exception</code>, <code>RuntimeException</code>, <code>Error</code> и их потомки. Различие между проверяемыми (checked) и непроверяемыми (unchecked) исключениями. Как работает блок <code>try-catch-finally</code>. Что произойдет, если исключение не перехвачено. Назначение ключевых слов <code>throw</code> и <code>throws</code>. Различие между ними. Порядок обработки нескольких <code>catch</code>-блоков. Как работает «вложенная» обработка. Что произойдет, если в блоке <code>finally</code> возникает новое исключение. Создание пользовательских исключений: правила наследования от <code>Exception</code> и реализация конструктора. Как логировать ошибки. Использование <code>printStackTrace()</code>, логгеров (<code>Logger</code>, <code>SLF4J</code>).	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
	Практическое применение обработки ошибок при чтении из файлов, делении на ноль и некорректном пользовательском вводе. Антипаттерны в обработке исключений: пустой catch, подавление исключений, избыточные throws.	
Коллекции данных и обобщения (Generics)	Назначение и преимущества использования коллекций в Java по сравнению с массивами. Отличия между List, Set и Map: семантика хранения, порядок элементов, уникальность. Классы ArrayList, LinkedList, HashSet, TreeSet, HashMap, TreeMap: внутренняя структура, особенности и применение. Итераторы: интерфейс Iterator, метод remove(), цикл for-each, fail-fast поведение. Введение в обобщенные типы (Generics): зачем они были введены, что такое параметризация типов. Синтаксис generics: объявление обобщенных классов, интерфейсов и методов. Ограничения обобщений в Java: type erasure, невозможность создать массив обобщенного типа. Связь generics с безопасностью типов и ранним обнаружением ошибок компиляции. Wildcards: ?, ? extends T, ? super T – зачем нужны и как используются. Принцип PECS (Producer Extends, Consumer Super).	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
	Как generics применяются в стандартных структурах данных и API (Collections, Comparator).	
Ввод-вывод в Java. Потоки ввода-вывода	Виды потоков в Java: байтовые и символьные, их иерархия и основные классы. Отличия InputStream/OutputStream и Reader/Writer, сферы применения. Buffered-потоки: для чего используются BufferedReader и BufferedWriter. File I/O: классы File, FileReader, FileWriter, PrintWriter. Современные возможности ввода-вывода: java.nio.file.*, классы Files, Paths, Path. Работа с кодировками: зачем нужно явно указывать Charset, чем отличается UTF-8 от Windows-1251. Потокобезопасность при работе с файлами. Типичные ошибки при работе с файлами: неправильное закрытие ресурсов, потеря данных. Конструкция try-with-resources: назначение и преимущества. Исключения, возникающие при работе с файлами: FileNotFoundException, IOException.	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.
Функциональное программирование в Java: лямбда-выражения и Stream API	Что такое функциональное программирование и как его элементы реализованы в Java? Понятие и назначение функциональных интерфейсов (Predicate, Consumer, Function, Supplier).	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
	Особенности синтаксиса лямбда-выражений: параметры, тело, возвращаемые значения. Область видимости переменных внутри лямбда-выражений. Понятие stream-потока (Stream<T>), создание потока из коллекции. Промежуточные и терминальные операции в Stream API: map, filter, sorted, collect, reduce, forEach. Понятие ленивых вычислений и неизменяемости в контексте Stream API. Параллельные потоки (parallelStream()) и особенности их применения. Чем лямбда-выражения отличаются от анонимных классов? Каковы плюсы и ограничения использования функционального стиля в Java?	
Аннотации и рефлексия	Что такое аннотации в Java? В чем отличие встроенных и пользовательских аннотаций? Основные стандартные аннотации Java: @Override, @Deprecated, @SuppressWarnings, их применение и назначение. Синтаксис объявления собственной аннотации, типы элементов в аннотации. Какие ограничения существуют при создании аннотаций? Как применять аннотации к классам, методам, полям?	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
	Что такое аннотации с retention policy SOURCE, CLASS, RUNTIME? Механизм работы рефлексии (Reflection API): получение информации о классах, методах, полях во время выполнения. Как создать экземпляр класса с помощью рефлексии? Безопасность при использовании Reflection API. Какие риски существуют? Как Java-фреймворки используют аннотации и рефлексия (пример: JUnit, Spring)?	
Модульная система Java	Что такое Java Platform Module System (JPMS) и зачем она была введена? Какие элементы включает модуль: module-info.java, requires, exports, opens? Чем exports отличается от opens? Как модульность влияет на инкапсуляцию и сборку проекта? Какова структура модульного проекта в Java? Как связаны модули и зависимые библиотеки? Как использовать jlink и jmod? Какие ошибки бывают в модульной системе (ModuleNotFoundException и др.) и как их устранять? Как происходит разрешение зависимостей между модулями?	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
	Как применять JPMS в IDE, например IntelliJ IDEA или Eclipse?	
Современные технологии корпоративной разработки на Java: Java EE и Spring	Какие компоненты входят в архитектуру Java EE / Jakarta EE? Что такое сервлеты и их жизненный цикл? Отличия Java EE и Spring Framework: подход к внедрению зависимостей, конфигурации, масштабируемости. Что такое IoC (Inversion of Control) и DI (Dependency Injection) в контексте Spring? Аннотации Spring: @Component, @Service, @Autowired, @RestController — назначение и сфера применения. Что такое Spring Boot и чем он отличается от классического Spring? Как работает автоконфигурация в Spring Boot? Что такое слои (controller, service, repository) и зачем они нужны? Как создать REST API с помощью Spring Boot? Понятие конфигурации приложения: application.properties, application.yml Подключение БД и работа с репозиториями через Spring Data JPA. Жизненный цикл Spring Bean'а и управление зависимостями.	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
Шаблоны проектирования в объектно-ориентированных системах	Что такое шаблон (паттерн) проектирования? Зачем они нужны? На какие категории делятся паттерны: порождающие, структурные, поведенческие? В чем отличие паттернов Factory Method и Abstract Factory? Когда следует применять паттерн Singleton? Его плюсы и проблемы. Что делает паттерн Observer? Примеры использования в UI и финансах. Как реализовать паттерн Strategy? Зачем отделять алгоритм от клиента? Что делает паттерн Decorator и как он помогает избегать подклассов? В чем суть SOLID-принципов? Как они связаны с паттернами? Как сочетать шаблоны в одном проекте (например, Singleton + Factory)? Чем паттерны отличаются от архитектурных стилей (MVC, MVVM)? Как документировать и обосновывать применение паттерна?	• работа с рекомендуемыми источниками литературы; • конспектирование; • выполнение практической работы.
Сравнение ООП-языков: Java, C++, C#, Python	Какие парадигмы реализуют Java, C++, C#, Python? В чем различие между статической и динамической типизацией? Как это влияет на проектирование?	• работа с рекомендуемыми источниками литературы;



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
	Как реализованы классы и объекты в C++, Java, Python и C#? Есть ли множественное наследование в этих языках? Как оно эмулируется? Что такое интерфейсы в Java и C#? Какую роль они играют? Как устроено управление памятью: сборка мусора против ручного контроля? Как реализованы конструкторы, деструкторы и финализаторы? Отличия в синтаксисе объявления классов и методов? Как выполняются программы: компиляция, байт-код, JIT-интерпретация? Что такое properties в C# и почему их нет в Java? Как реализуется перегрузка и переопределение методов? Поддержка функционального программирования: лямбды, замыкания В каких областях используется каждый из языков? Где Java и где Python предпочтительнее? Что такое duck typing в Python? Как реализуются исключения, какие существуют типы ошибок?	• конспектирование; • выполнение практической работы.
Юнит-тестирование и обеспечение качества кода	Что такое юнит-тестирование? Чем оно отличается от интеграционного и системного?	• работа с рекомендуемыми источниками литературы; • конспектирование;



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
	Как организовать структуру тестов в Java с использованием JUnit 5? Что такое test case, test suite и test runner? Какие аннотации используются в JUnit 5 (@Test, @BeforeEach, @AfterEach, @Disabled)? Что такое TDD (Test Driven Development) и в чем его преимущества? Что такое mock-объекты и зачем они нужны в тестировании? Как измеряется покрытие кода тестами (code coverage)? Почему тесты помогают в рефакторинге? Что такое анти-паттерны в тестировании? Как логирование (Log4j, SLF4J) помогает выявлять ошибки? Различие между позитивными и негативными тестами В чем разница между assertEquals, assertThrows и assertTimeout? Почему важно тестировать исключения? Как тестировать методы с внешними зависимостями (БД, файлы)? Что делать, если тесты "хрупкие" и часто падают?	• выполнение практической работы.



6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю

Примерная тематика контрольной работы

1. Информационная система «Учебный портал»
2. Реализовать классы: Student, Course, Enrollment
3. Возможности: регистрация студентов, запись на курсы, выставление оценок, расчет среднего балла
4. ООП: наследование (OnlineCourse и OnCampusCourse), интерфейсы (Gradable, Printable)
5. Использование коллекций (Map, List)
6. Отчет в формате CSV
7. Тестирование с использованием JUnit
8. Финансовый менеджер (Personal Finance Tracker)
9. Классы: Transaction, Category, Wallet
10. Функции: добавление расходов/доходов, сортировка, фильтрация, баланс по категориям
11. ООП: полиморфизм (ExpenseTransaction / IncomeTransaction), шаблон Factory
12. Сериализация данных в JSON
13. UI через консоль или JavaFX (по желанию)
14. Микросистема бронирования ресурсов
15. Классы: Room, Booking, User
16. Возможности: создание бронирования, отмена, просмотр расписания
17. Использование шаблона Observer: уведомление о подтверждении/отмене
18. Тестирование всех методов бронирования
19. Документирование проекта: UML-диаграммы, README
20. Консольная игра «Угадай слово» с графическим отчетом
21. Разработка на основе ООП-архитектуры: Game, Player, WordGenerator
22. Статистика побед/поражений сохраняется в файл
23. Использование Enum, Stream API
24. Внедрение шаблона Strategy: разные уровни сложности
25. REST API-сервер (Spring Boot, опционально)
26. Темы: REST-интерфейс к системе учета задач (ToDo)
27. Применение Spring: @RestController, @Service, @Repository
28. Разделение на слои, применение IoC и DI
29. Уровень сложности выше среднего



Дополнительная информация

Критерии балльной оценки различных форм текущего контроля успеваемости содержатся в соответствующих методических рекомендациях Кафедры информационных технологий Факультета информационных технологий и анализа больших данных.



7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине

Перечень компетенций с указанием индикаторов их достижения в процессе освоения образовательной программы содержится в разделе 2. «Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине».

Типовые контрольные задания или иные материалы, необходимые для оценки индикаторов достижения компетенций, умений и знаний

Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
ПКП-5. Способен разрабатывать требования и проектировать программное обеспечение	Анализирует возможности реализации требований к компьютерному программному обеспечению	Знать: методы анализа и формализации требований к ПО; основные подходы к проектированию программного обеспечения; Уметь: анализировать и формализовывать требования; оценивать их реализуемость;	1. Теоретические вопросы: • Объяснить различие между функциональными и нефункциональными требованиями. • Охарактеризовать методы анализа и формализации требований к программному обеспечению. 2. Практическое задание: • По описанию предметной области: • выделить и сформулировать требования к системе; • разделить требования на функциональные и нефункциональные; • выявить недостающие, противоречивые или некорректные требования; • оценить реализуемость требований; • предложить уточненную формулировку требований. 3. Задание на анализ: • Проанализировать предложенный набор требований;



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
			• выявить ошибки, противоречия и неоднозначности; • определить, какие требования невозможно или затруднительно реализовать в предложенных условиях; • предложить улучшения и обосновать их. 4. Мини-проект: По словесному описанию работы информационной системы сформировать перечень требований, оценить их полноту и реализуемость, указать возможные риски реализации.
	Проектирует компьютерное программное обеспечение	Знать: архитектурные стили и шаблоны проектирования; принципы ООП и декомпозиции систем; Уметь: проектировать архитектуру и структуру ПО; разрабатывать модели системы и обосновывать решения;	1. Теоретические вопросы: • Объяснить назначение шаблонов проектирования в объектно-ориентированных системах. • Охарактеризовать роль принципов ООП и декомпозиции при проектировании структуры программной системы. 2. Практическое задание: • По описанию предметной области: • спроектировать архитектуру программной системы; • выделить основные компоненты или подсистемы; • предложить структуру классов или модулей; • представить решение в виде UML-диаграммы или структурной схемы; • обосновать выбранные архитектурные решения и примененные шаблоны проектирования.



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
			3. Задание на анализ: • Проанализировать предложенное архитектурное решение; • выявить недостатки в декомпозиции системы и взаимодействии компонентов; • определить, насколько корректно применены принципы ООП и шаблоны проектирования; • предложить улучшения и обосновать их. 4. Мини-проект: • Разработать проект программной системы для выбранной предметной области; • определить назначение системы и ее основные функции; • выделить основные сущности, компоненты и связи между ними; • спроектировать архитектуру и структуру классов; • подготовить краткое описание компонентов, их функций и взаимодействия.
ПКП-8. Способен выполнять работы и управлять работами по созданию (модификации) и сопровождению информа-	Разрабатывает модели бизнес-процессов заказчика в рамках проекта создания (модификации) ИС	Знать: методы моделирования бизнес-процессов (BPMN, UML); основы анализа предметной области и требований заказчика; Уметь: моделировать бизнес-процессы за-	1. Теоретические вопросы: • Описать основные этапы проектирования программного обеспечения. • Привести примеры архитектурных стилей и объяснить области их применения. 2. Практическое задание: • По описанию предметной области: • спроектировать архитектуру программной системы; • выделить основные компоненты и описать их взаимодействие; 3. Задание на анализ: • Проанализировать предложенное архитектурное решение;



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
ционных систем (ИС), автоматизирующих задачи организационного управления и бизнес-процессы		казчика; анализировать требования и формализовывать их;	• выявить недостатки в декомпозиции системы и взаимодействии компонентов; 4. Мини-проект: Разработать проект программной системы: определить основные компоненты, построить модель системы, описать архитектуру и ключевые связи между компонентами, обосновать выбранные проектные решения.
	Разрабатывает архитектуру ИС в рамках выполнения работ и управления работами по созданию (модификации) и сопровождению ИС	Знать: принципы построения архитектуры информационных систем; типовые архитектурные решения ИС; Уметь: проектировать архитектуру информационной системы; выбирать и обосновывать архитектурные решения;	1. Теоретические вопросы: • Объяснить роль шаблонов проектирования при разработке программного обеспечения. • Охарактеризовать принципы ООП и декомпозиции систем при проектировании программных решений. 2. Практическое задание: • предложить структуру классов или модулей; • представить решение в виде UML-диаграммы или структурной схемы; • обосновать выбор архитектурного решения и используемых шаблонов проектирования. 3. Задание на анализ: • определить, насколько корректно применены архитектурные подходы и шаблоны проектирования; • предложить улучшения и обосновать их. 4. Мини-проект: • разработать проект программного обеспечения для заданной предметной области;



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
			• определить основные сущности и связи между ними; • предложить структуру классов или модулей; • подготовить краткое описание структуры системы и взаимодействия ее компонентов.



Примеры практико-ориентированных заданий

1. Реализовать класс иерархию геометрических фигур (круг, прямоугольник, треугольник), где базовый класс содержит абстрактный метод вычисления площади, а подклассы реализуют его по-своему. Проверяется понимание наследования и полиморфизма.
2. Написать метод, который на вход принимает список целых чисел и возвращает новый список, содержащий только уникальные элементы исходного (использовать коллекцию Set). Оценить сложность решения.
3. Применить шаблон проектирования «Наблюдатель»: создать класс наблюдателя, реагирующего на изменение состояния другого объекта (например, при обновлении курса валют наблюдатель выводит уведомление).

Примеры вопросов для подготовки к промежуточной аттестации

Примерные вопросы для подготовки к зачету

1. Основные принципы объектно-ориентированного программирования: инкапсуляция, наследование, полиморфизм.
2. Классы и объекты в языке Java. Определение, структура, жизненный цикл.
3. Перегрузка и переопределение методов. Отличия и примеры.
4. Конструкторы и модификаторы доступа в Java.
5. Абстрактные классы и интерфейсы. Сравнительный анализ и область применения.
6. Ключевые слова this, super, final и их роль в ООП-модели Java.
7. Обработка исключений в Java: механизм, иерархия, создание пользовательских исключений.
8. Стандартные коллекции Java: List, Set, Map. Примеры и области применения.
9. Обобщённые типы (generics) в Java. Мотивация, синтаксис, ограничения.
10. Структура и особенности ArrayList, LinkedList, HashMap, TreeMap.
11. Механизм автоупаковки и анбоксинга (autoboxing/unboxing).
12. Потоки ввода-вывода в Java: работа с файлами, сериализация.
13. Классы BufferedReader, FileReader, Files, Paths: возможности и различия.
14. Лямбда-выражения и функциональные интерфейсы: синтаксис и назначение.
15. Stream API: архитектура, основные операции, примеры использования.
16. Модульная система Java (JPMS): структура, особенности, module-info.java.
17. Структура Java-программы. Жизненный цикл исполнения.
18. Модификаторы доступа и область видимости в Java.



19. Шаблоны проектирования: Singleton, Factory, Observer и их реализация на Java.
20. Понятие SOLID-принципов и их реализация в Java.
21. Понятие и реализация принципа Dependency Injection (DI) в Spring Framework.
22. IoC-контейнер и аннотации в Spring: @Component, @Service, @Autowired.
23. Основы Spring Boot: структура проекта, автоконфигурация, аннотации.
24. Тестирование с использованием JUnit 5: структура теста, основные аннотации.
25. Жизненный цикл unit-тестов и концепция TDD (Test-Driven Development).
26. Параметризованные тесты в JUnit. Валидация и тест-кейсы.
27. Java vs C++: управление памятью, сборка, реализация ООП.
28. Java vs Python: строгость типизации, модель исполнения, парадигмы.
29. Сравнение Java и C#: синтаксис, свойства (properties), .NET vs JVM.
30. Использование аннотаций и механизм рефлексии в Java.
31. Проектирование и реализация мини-приложений с применением ООП.
32. Архитектура многомодульного проекта в Java.
33. Практика использования коллекций и дженериков в реальных задачах.
34. Роль интерфейсов функционального программирования в современном Java-коде.
35. Типичные ошибки при реализации ООП в Java и способы их предотвращения.



8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины

Основная литература:

1. Крючкова, Е. Н. Объектно-ориентированное программирование: Архитектурное проектирование и паттерны программирования [Электронный ресурс] : учебно-методическое пособие для студентов направления 09.03.04 «программная инженерия» / Крючкова Е. Н., Старолетов С. М. Барнаул : АлтГТУ, 2020 180 с. Книга из коллекции АлтГТУ - Информатика <https://e.lanbook.com/book/292790>. [БИК ID: RU-LAN-BOOK-292790]

Дополнительная литература:

1. Шнейдеров, Е. Н. Разработка приложений на языке Java. Лабораторный практикум [Электронный ресурс] : пособие / Шнейдеров Е. Н., Писарчик А. Ю., Казючиц В. О. БГУИР : БГУИР, 2023 92 с. Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве пособия для специальности 1-39 03 02 «Программируемые мобильные системы» Книга из коллекции БГУИР - Информатика <https://e.lanbook.com/book/479516> ISBN 978-985-543-561-8. [БИК ID: RU-LAN-BOOK-479516]
2. Введение в современную Android-разработку на языке Java : учебное пособие / Рысин М. Л. Ч. 1 : Введение в современную Android-разработку на языке Java. Часть 1 : учебное пособие . Ч. 1 / Рысин М. Л. Москва : РТУ МИРЭА, 2023 132 с. Книга из коллекции РТУ МИРЭА - Информатика <https://e.lanbook.com/book/382586> ISBN 978-5-7339-1895-2. [БИК ID: RU-LAN-BOOK-382586]
3. Бобырь, Максим Владимирович Программирование на языке Java. Практический курс : Учебное пособие / Юго-Западный государственный университет 1 Москва : ООО "Научно-издательский центр ИНФРА-М", 2025 189 с. (Высшее образование) ВО - Бакалавриат <https://znanium.ru/catalog/document?id=460819> ISBN 978-5-16-020136-8 ISBN 978-5-16-112676-9 (электр. издание) . [БИК ID: RU\infra-m\znanium\bibl\2160989]
4. Введение в современную Android-разработку на языке Java / Рысин М. Л. Ч. 2 : Введение в современную Android-разработку на языке Java. Часть 2 : учебное пособие . Ч. 2 / Рысин М. Л. Москва : РТУ МИРЭА, 2024 110 с. Книга из коллекции РТУ МИРЭА - Информатика



<https://e.lanbook.com/book/420962> ISBN 978-5-7339-2146-4. [БИК ID: RU-LAN-BOOK-420962]

5. Коротеев, М.В. Введение в Android разработку на Java : Учебное пособие / М.В. Коротеев, А.Ю. Шаталова Электрон. дан. Москва : КноРус, 2026 231 с. Режим доступа: book.ru Internet access <https://book.ru/book/960287> ISBN 978-5-406-15558-5. [БИК ID: RU\bookru\bibl\960287]

Нормативные правовые акты:

-

9. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/> (<http://library.fa.ru/files/elibfa.pdf>)
2. Электронно-библиотечная система издательства "Лань" <https://e.lanbook.com/>
3. • Научная электронная библиотека eLibrary.ru <http://elibrary.ru>
4. Информационно-образовательный портал Финуниверситета: <https://org.fa.ru>



10. Методические указания для обучающихся по освоению дисциплины

Общие рекомендации по изучению курса: Дисциплина сочетает теоретический материал и значительный практический компонент. Рекомендуется активно участвовать в лекциях – обращать внимание на основные определения (например, принципы ООП, структуры данных, шаблоны проектирования) и примеры кода, задавать вопросы по неясным моментам. После лекции полезно просматривать конспект и сверяться с рекомендованным учебником для углубления понимания.

На практических занятиях (семинарах) основная задача – научиться применять теорию на практике: писать код, разбирать ошибки, обсуждать варианты реализации. Для эффективного освоения, перед каждым семинаром студенту стоит повторно изучить тему лекции и попытаться выполнить базовые упражнения самостоятельно. Например, перед занятием по коллекциям – потренироваться создавать список и выполнять над ним операции. Не бойтесь ошибаться при написании кода во время занятий: каждое обнаруженное и исправленное ошибка – шаг к лучшему пониманию языка. Полезно брать с собой ноутбук либо заранее готовить шаблоны проектов, чтобы на семинаре фокусироваться на логике решения задач.

Самостоятельная работа играет большую роль в овладении навыками программирования. В рамках данной дисциплины предусмотрен курсовой мини-проект – начать работу над ним следует задолго до дедлайна. Разбейте проект на этапы: (1) постановка задачи и проектирование (нарисуйте схему классов, спланируйте функциональность), (2) реализация основных классов и методов, (3) отладка и тестирование, (4) рефакторинг и улучшение структуры, (5) написание отчета и подготовка презентации (если требуется). Регулярно показывайте промежуточный результат преподавателю или наставнику – это поможет направить работу в правильное русло. Также полезно сверяться с профессиональными требованиями.

Использование ресурсов: В процессе обучения настоятельно рекомендуется обращаться к документации и сообществам. Официальная JavaDoc – ваш главный справочник по классам и методам: привыкайте искать там информацию о незнакомых функциях. При возникновении трудностей с кодом – формулируйте проблему и ищите решение (в литературе или на доверенных форумах, таких как StackOverflow). Однако важно придерживаться академической добросовестности: если вы заимствуете фрагменты кода или идеи из внешних источников, старайтесь их понять, прокомментировать и не копировать без разбора.

Связь с профессиональной деятельностью: Освоение данной дисциплины подготовит вас к реальным задачам в индустрии. Навыки ООП и владение Java



требуются программистам в разработке самых различных приложений – от мобильных до корпоративных. Поэтому относитесь к заданиям серьезно: то, как вы научитесь структурировать код и работать в команде над проектом сейчас, повлияет на вашу эффективность как будущего специалиста. Развивайте не только навыки кодирования, но и soft skills: работу с источниками информации, умение задавать вопросы, сотрудничать с коллегами (например, в групповых обсуждениях на семинарах).

Консультации и обратная связь: При затруднениях не стесняйтесь обращаться к преподавателю на консультациях. Лучше задать вопрос и разобраться, чем оставаться в неопределенности. Преподаватель может подсказать, где найти нужную информацию или как правильнее организовать решение. Также полезно обмениваться опытом с одногруппниками: обсуждение решений (не списывание, а именно обмен идеями!) часто помогает увидеть иную точку зрения и улучшить свой код.

При подготовке контрольной работы следует использовать нормативные документы Финансового университета, Методические рекомендации по планированию и организации внеаудиторной самостоятельной работы студентов по образовательным программам бакалавриата и магистратуры в Финансовом университете, утвержденные приказом Финуниверситета от 11.05.2021 г. № 1040/о (см. сайт Финансового Университета: на главной странице раздел "Наш университет" → "Единая правовая база Финуниверситета" → "Организация учебного процесса" → "Нормативные документы по самостоятельной работе").



11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем

Комплект лицензионного программного обеспечения:

1. Операционная система
2. Текстовый редактор
3. Пакет офисных программ
4. Антивирус Kaspersky

Современные профессиональные базы данных и информационные справочные системы

1. Информационно-правовая система «Гарант»
2. Информационно-правовая система «Консультант Плюс»
3. Электронная энциклопедия: <http://ru.wikipedia.org/wiki/Wiki>
4. Система комплексного раскрытия информации «СКРИН» - <http://www.skrin.ru/>

Сертифицированные программные и аппаратные средства защиты информации

1. не предусмотрены

12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

1. **Компьютерный класс** для проведения учебных занятий, предусмотренных программой, в том числе групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, оснащенный оборудованием и техническими средствами обучения: мебель аудиторная (столы, стулья, доска аудиторная), персональные компьютеры, набор демонстрационного оборудования (проектор, экран)
2. Библиотека, читальный зал
3. Учебная аудитория для проведения учебных занятий, предусмотренных программой, в том числе групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, оснащенная оборудованием и техническими средствами обучения: мебель аудиторная (столы, стулья, доска аудиторная), набор демонстрационного оборудования (проектор, экран)