МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет компьютерных технологий и прикладной математики

УТВЕРЖДАЮ
Проректор по учебной работе,
качеству образования – первый
проректор
_____ Хагуров Т.А.
подпись
«02» июня 2026

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.О.24 Объектно-ориентированное программирование

Направление подготовки 02.03.03 Математическое обеспечение и
администрирование информационных систем

Направленность (профиль) Искусственный интеллект и аналитика данных

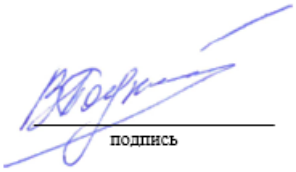
Форма обучения очная

Квалификация бакалавр

Краснодар 2026

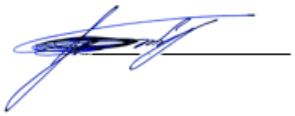
Рабочая программа дисциплины «Объектно-ориентированное программирование» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования (ФГОС ВО) по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем.

Программу составил(и):
Подколзин В.В. канд. физ.-мат. наук, доцент
И.О. Фамилия, должность, ученая степень, ученое звание

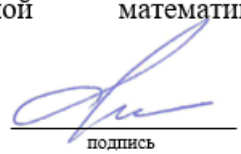


подпись

Рабочая программа дисциплины утверждена на заседании центра искусственного интеллекта
протокол № 04 «14» мая 2026 г.
Руководитель центра ИИ Коваленко А.В.



Утверждена на заседании учебно-методической комиссии факультета компьютерных технологий и прикладной математики
протокол №3 от «15» мая 2026.
Председатель УМК факультета
Добровольская Н.Ю.



подпись

Рецензенты:
Мостовой Евгений Викторович, генеральный директор ООО «Портал-Юг», e-mail: mostovoy@portal-yug.ru

Луценко Евгений Вениаминович, доктор экономических наук, кандидат технических наук, профессор кафедры компьютерных технологий и систем Федерального государственного бюджетное образовательное учреждение высшего образования «Кубанский государственный аграрный университет имени И.Т. Трубилина», e-mail: prof.lutsenko@gmail.com

1 Цели и задачи изучения дисциплины (модуля)

1.1 Цель освоения дисциплины

Цель дисциплины – изучение методологии объектно-ориентированного программирования.

1.2 Задачи дисциплины

1. Изучение подхода объектно-ориентированного программирования на примере языка Java.
2. Изучение основных принципов ООП и подходов к разработке объектной модели.
3. Изучение ИИ-инструменты в разработке: генерация кода и тестов.
4. Получение практического опыта в применении ООП к задачам ИИ.
5. Получение практического опыта в работе с современными IDE платформами.

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Объектно-ориентированное программирование» относится к дисциплинам базовой части, код Б1.О.24.

Дисциплина в значительной степени **взаимодействует для формирования компетенций** с дисциплинами:

1. Программирование
2. Алгоритмы и структуры данных
3. Фундаментальные дискретные модели
4. Промпт-инжиниринг

Требованием к «входным» знаниям является понимание основ программирования на императивных языках, методов моделирования и основ работы с ИИ.

1.4 Профессиональные роли в структуре образовательной программы

Роль 1: Data Engineer (Инженер по данным)
Задачи:

- Проектирование и построение ETL-процессов
- Создание и оптимизация хранилищ данных
- Обеспечение качества и доступности данных
- Настройка инфраструктуры для обработки больших данных
- Интеграция разрозненных источников данных
- Работа с данными в области природопользования, медицины, связи и телекоммуникаций

Роль 2: AI Architect (Архитектор ИИ)
Задачи:

- Проектирование архитектуры ИИ-систем
- Выбор технологий и инструментов для ИИ-проектов
- Интеграция ИИ-компонентов в корпоративные системы
- Обеспечение масштабируемости и производительности
- Техническое руководство ИИ-проектами

Роль 3: MLOps (Специалист по эксплуатации ИИ)
Задачи:

- Автоматизация процессов обучения и развертывания моделей
- Мониторинг производительности ML-систем
- Управление версиями моделей и данных

- Обеспечение CI/CD для ML-проектов
- Оптимизация вычислительных ресурсов

1.5 Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Изучение данной учебной дисциплины направлено на формирование у обучающихся следующих компетенций:

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 Аргументировано применяет современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения

ОПК-3.2 Ориентируется в современных положениях и концепциях прикладного и системного программного обеспечения, архитектуры компьютеров и сетей (в том числе и глобальных), технологии создания и сопровождения программных продуктов и программных комплексов

PL-2 ***Способен применять JVM-совместимые языки программирования для решения задач в области ИИ***

PL-2.1 *Разрабатывает и отлаживает прикладные решения разного уровня сложности и для широкого круга конечных пользователей с использованием JVM-совместимых языков программирования, тестирует, испытывает и оценивает качество таких решений*

Значимость компетенций для каждой из трёх профессиональных ролей.

Роль	**Ключевые задачи роли**	**Соответствующие компетенции (с указанием индикаторов)**
Data Engineer (Инженер по данным)	– Проектирование и построение ETL-процессов – Создание и оптимизация хранилищ данных – Обеспечение качества и доступности данных – Настройка инфраструктуры для обработки больших данных – Интеграция разрозненных источников данных – Работа с данными в прикладных областях	**ОПК-3.1** – применение современных информационных технологий (включая отечественные) при создании программных продуктов (ETL, хранилища) **ОПК-3.2** – ориентация в концепциях прикладного и системного ПО, архитектуре компьютеров и сетей (важно для распределённой обработки данных) **PL-2.1** – разработка и отладка прикладных решений на JVM-языках, тестирование (для реализации ETL-модулей на Java)

Роль	Ключевые задачи роли	Соответствующие компетенции (с указанием индикаторов)
AI Architect (Архитектор ИИ)	– Проектирование архитектуры ИИ-систем – Выбор технологий и инструментов для ИИ-проектов – Интеграция ИИ-компонентов в корпоративные системы – Обеспечение масштабируемости и производительности – Техническое руководство ИИ-проектами	**ОПК-3.1** – аргументированное применение современных ИТ, в том числе отечественных (например, YandexGPT, GigaChat) при создании ИИ-продуктов **ОПК-3.2** – знание концепций системного ПО, архитектуры компьютеров и сетей (для выбора оптимальной инфраструктуры ИИ) **PL-2.1** – разработка и отладка прикладных решений на JVM-языках, проектирование иерархий классов, модульное тестирование, цикл «промпт → валидация → интеграция»
MLOps (Специалист по эксплуатации ИИ)	– Автоматизация процессов обучения и развертывания моделей – Мониторинг производительности ML-систем – Управление версиями моделей и данных – Обеспечение CI/CD для ML-проектов – Оптимизация вычислительных ресурсов	**ОПК-3.1** – применение современных информационных технологий (CI/CD, контейнеризация, оркестрация) при создании программных комплексов **ОПК-3.2** – ориентация в технологиях создания и сопровождения программных продуктов (необходима для настройки пайплайнов) **PL-2.1** – разработка и отладка прикладных решений на JVM-языках, тестирование, критическая проверка и доработка кода, сгенерированного ИИ (для автоматизации развёртывания моделей)

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 3 зач. ед. (108 часов), их распределение по видам работ представлено в таблице

Вид учебной работы	Всего часов	Семестры (часы)					
		3					
Контактная работа, в том числе:	**54,3**	**54,3**					
Аудиторные занятия (всего):	**50**	**50**					

Занятия лекционного типа		**16**	16					
Лабораторные занятия		**34**	34					
Занятия семинарского типа (семинары, практические занятия)								
Иная контактная работа:		**2,3**	**2,3**					
Контроль самостоятельной работы (КСР)		**2**	2					
Промежуточная аттестация (ИКР)		**0,3**	0,3					
Самостоятельная работа, в том числе:		**20**	**20**					
Курсовая работа								
Выполнение индивидуальных заданий		**20**	20					
Реферат								
Подготовка к текущему контролю								
Контроль:		**35,7**	**35,7**					
Подготовка к экзамену		**35,7**	35,7					
Общая трудоемкость	**час.**	**108**	**108**					
	в том числе контактная работа	**54,3**	**54,3**					
	зач. ед	**3**	**3**					

2.2 Структура дисциплины

Распределение видов учебной работы и их трудоемкости по разделам дисциплины.

Разделы (темы) дисциплины, изучаемые в 3 семестре

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1	2	3	4	5	6	7
1.	Введение в платформу Java и среду разработки	6	2		2	2
2.	Базовые конструкции языка	6	2		2	2
3.	Классы и объекты	24	6		12	6
4.	Исключения	5	1		2	2
5.	Коллекции и обобщения	8	2		4	2
6.	Основы юнит-тестирования (JUnit)	7	1		4	2
7.	ИИ-инструменты в разработке: генерация кода и тестов	12	2		8	2
ИТОГО по разделам дисциплины		**70**	**16**		**34**	**20**
Контроль самостоятельной работы (КСР)		2				
Промежуточная аттестация (ИКР)		0,3				
Подготовка к текущему контролю		35,7				
Общая трудоемкость по дисциплине		**108**				

Примечание: Л – лекции, ПЗ – практические занятия/семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

2.3 Содержание разделов (тем) дисциплины

2.3.1 Занятия лекционного типа

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
1.	Введение в платформу Java и среду разработки	JDK, JRE, JVM. Установка IntelliJ IDEA Community. Простейшая программа. Типы данных, переменные, операторы.	ЛР
2.	Базовые конструкции языка	Условные операторы, циклы, массивы. Оформление кода, комментарии.	ЛР
3.	Классы и объекты. Инкапсуляция	Поля, методы, конструкторы. Перегрузка методов. Модификаторы доступа, пакеты, принцип сокрытия данных.	ЛР
4.	Наследование и полиморфизм	extends, super, переопределение методов, динамическое связывание. Полиморфизм через наследование.	ЛР
5.	Абстрактные классы и интерфейсы	Абстрактные методы, разница между интерфейсом и абстрактным классом, множественная реализация.	ЛР
6.	Исключения	try-catch-finally, throws, создание собственных исключений.	ЛР
7.	Коллекции и обобщения (Generics)	List, Set, Map. Итераторы. Параметризованные классы и методы	ЛР
8.	Основы юнит-тестирования (JUnit)	Аннотации @Test, @BeforeEach, утверждения assertEquals, assertThrows.	ЛР
9.	ИИ-инструменты в разработке: генерация кода и тестов	Обзор YandexGPT/GigaChat. Формулирование промптов. Генерация реализации по спецификации, рефакторинг, создание unit-тестов. Этика и ограничения.	ЛР

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

Соответствие тем лекций компетенциям

№	Тема лекции	Формируемые компетенции	Обоснование
1	Введение в платформу Java и среду разработки (JDK, JRE, JVM, IntelliJ IDEA)	ОПК-3.2, PL-2.1	Знание архитектуры JVM, инструментов – база для системного ПО; первые шаги в Java.
2	Базовые конструкции языка (условия, циклы, массивы)	ОПК-3.2, PL-2.1	Основы алгоритмизации и структур данных (концепции информатики), написание простейших программ.
3	Классы и объекты. Инкапсуляция	ОПК-3.1, ОПК-3.2, PL-2.1	Применение ООП при создании ПО (ОПК-3.1), понимание концепций классов (ОПК-3.2),

№	Тема лекции	Формируемые компетенции	Обоснование
			реализация на Java (PL-2.1).
4	Наследование и полиморфизм	ОПК-3.1, ОПК-3.2, PL-2.1	Построение иерархий – основа масштабируемых программ; полиморфизм – ключевая концепция ООП.
5	Абстрактные классы и интерфейсы	ОПК-3.1, ОПК-3.2, PL-2.1	Проектирование контрактов, множественная реализация – современные технологии создания ПО.
6	Исключения (try-catch-finally, throws, свои исключения)	ОПК-3.2, PL-2.1	Концепция обработки ошибок в системном ПО, реализация на Java.
7	Коллекции и обобщения (Generics)	ОПК-3.2, PL-2.1	Фундаментальные структуры данных (List, Set, Map) и их параметризация – важная часть концепций информатики.
8	Основы юнит-тестирования (JUnit)	ОПК-3.1, PL-2.1	Применение современных технологий тестирования (JUnit) для контроля качества ПО (ОПК-3.1); навык написания тестов (PL-2.1).
9	ИИ-инструменты в разработке (YandexGPT/GigaChat, промпты, генерация кода и тестов)	ОПК-3.1, PL-2.1	Использование отечественных ИИ-сервисов (ОПК-3.1); полный цикл «промпт → валидация → интеграция» (PL-2.1).

2.3.2 Занятия семинарского типа

Не предусмотрены

2.3.3 Лабораторные занятия

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4
1.	Установка инструментов. Hello World (2 часа)	Установить JDK 17+, IntelliJ IDEA. Написать и запустить программу «Привет, мир». Освоение отладчика.	ЛР

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4
2.	Управляющие конструкции и массивы (2 часа)	адачи на условия, циклы, обработку одномерных/двумерных массивов (без ООП).	ЛР
3.	Классы и объекты. Инкапсуляция (2 часа)	Разработать класс «Студент» с полями, конструкторами, методами доступа. Создание объектов, демонстрация инкапсуляции.	ЛР
4.	Наследование и полиморфизм (4 часа)	Иерархия классов «Фигура» → «Круг», «Прямоугольник». Полиморфный вызов методов площади.	ЛР
5.	Абстрактные классы и интерфейсы (6 час.)	Реализация интерфейса «Оплата» для нескольких способов (карта, наличные). Использование абстрактного класса.	ЛР
6.	Исключения (2 часа)	Модифицировать предыдущую задачу: генерация и обработка исключений (недостаточно средств и т.п.). Сгенерировать код обработчиков с помощью ИИ.	ЛР
7.	Генерация кода и модульных тестов с ИИ (4 часа)	По текстовому описанию предметной области (например, «Библиотека») получить от YandexGPT каркас классов и JUnit-тесты. Проверить, доработать, добиться прохождения тестов.	ЛР
8.	Коллекции и Generics (2 часа)	Реализовать простой репозиторий (List/Map) с параметризованными методами. Сгенерировать методы поиска и фильтрации через ИИ.	
9.	Работа с файлами (ввод-вывод) (2 часа)	Сериализация/десериализация объектов в JSON (библиотека org.json**).** ИИ генерирует шаблон кода для чтения/записи.	ЛР
10.	Итоговый мини-проект с обязательным применением ИИ (4 часа)	Командная/индивидуальная разработка небольшого приложения (например, «Заметки с категориями**»).** С помощью ИИ: генерация архитектуры классов, реализация отдельных модулей, создание полного набора JUnit-тестов. Защита — демонстрация работающего кода и пройденных тестов.	ЛР

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

Соответствие тем лабораторных занятий компетенциям

№	Название лабораторной работы	Формируемые компетенции	Обоснование
1	Установка инструментов. Hello World (JDK, IntelliJ IDEA, отладчик)	ОПК-3.2, PL-2.1	Освоение среды разработки и базового синтаксиса.
2	Управляющие конструкции и массивы (калькулятор, матрица)	ОПК-3.2, PL-2.1	Реализация алгоритмов без ООП – закрепление концепций.
3	Классы и объекты. Инкапсуляция (класс «Студент», «Время»)	ОПК-3.1, ОПК-3.2, PL-2.1	Создание классов с полями, методами, валидацией – приложение ООП на практике.
4	Наследование и полиморфизм (фигуры, транспорт)	ОПК-3.1, ОПК-3.2, PL-2.1	Построение иерархий, полиморфные вызовы – ключевой навык ООП.
5	Абстрактные классы и интерфейсы (платежи, сортировка)	ОПК-3.1, ОПК-3.2, PL-2.1	Реализация интерфейсов, абстракций – основа модульности и тестируемости.
6	Исключения (генерация через ИИ кастомных исключений)	ОПК-3.2, PL-2.1	Первое применение ИИ для генерации кода; обработка ошибок.
7	Генерация кода и модульных тестов с ИИ (модель «Библиотека», CRUD-репозиторий)	ОПК-3.1, PL-2.1	Полный цикл работы с ИИ (промпты → код → тесты → доработка) с использованием отечественных сервисов.
8	Коллекции и Generics (контейнер Box, статистика текста)	ОПК-3.2, PL-2.1	Применение параметризованных коллекций, алгоритмов фильтрации – концепции структур данных.
9	Работа с файлами (сериализация/десериализация JSON, логирование)	ОПК-3.2, PL-2.1	Ввод-вывод, работа с форматами данных – важная часть системного ПО.
10	Итоговый мини-проект с обязательным применением ИИ (Менеджер задач / Заметки с категориями)	ОПК-3.1, ОПК-3.2, PL-2.1	Комплексная разработка: архитектура, ООП, тесты, ИИ-генерация, сериализация – все три компетенции интегрированы.

2.3.4 Примерная тематика курсовых работ (проектов)

Курсовая работа не предусмотрена.

2.4 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

№	Вид СРС	Перечень учебно-методического обеспечения дисциплины по выполнению самостоятельной работы
1	2	3
1	Изучение теоретического материала	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой информационных технологий, протокол №1 от 30.08.2019
2	Решение задач	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой информационных технологий, протокол №1 от 30.08.2019

Учебно-методические материалы для самостоятельной работы обучающихся из числа инвалидов и лиц с ограниченными возможностями здоровья (ОВЗ) предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа,

– в форме аудиофайла,

– в печатной форме на языке Брайля.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа,

– в форме аудиофайла.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

3. Образовательные технологии

В соответствии с требованиями ФГОС в программа дисциплины предусматривает использование в учебном процессе следующих образовательные технологии: чтение лекций с использованием мультимедийных технологий; метод малых групп, разбор практических задач и кейсов.

При обучении используются следующие образовательные технологии:

- Технология коммуникативного обучения – направлена на формирование коммуникативной компетентности студентов, которая является базовой, необходимой для адаптации к современным условиям межкультурной коммуникации.
- Технология разноуровневого (дифференцированного) обучения – предполагает осуществление познавательной деятельности студентов с учётом их индивидуальных способностей, возможностей и интересов, поощряя их реализовывать свой творческий потенциал. Создание и использование диагностических тестов является неотъемлемой частью данной технологии.
- Технология модульного обучения – предусматривает деление содержания дисциплины на достаточно автономные разделы (модули), интегрированные в общий курс.
- Информационно-коммуникационные технологии (ИКТ) - расширяют рамки образовательного процесса, повышая его практическую направленность, способствуют

интенсификации самостоятельной работы учащихся и повышению познавательной активности. В рамках ИКТ выделяются 2 вида технологий:

- Технология использования компьютерных программ – позволяет эффективно дополнить процесс обучения языку на всех уровнях.
- Интернет-технологии – предоставляют широкие возможности для поиска информации, разработки научных проектов, ведения научных исследований.
- Технология индивидуализации обучения – помогает реализовывать личностно-ориентированный подход, учитывая индивидуальные особенности и потребности учащихся.
- Проектная технология – ориентирована на моделирование социального взаимодействия учащихся с целью решения задачи, которая определяется в рамках профессиональной подготовки, выделяя ту или иную предметную область.
- Технология обучения в сотрудничестве – реализует идею взаимного обучения, осуществляя как индивидуальную, так и коллективную ответственность за решение учебных задач.
- Игровая технология – позволяет развивать навыки рассмотрения ряда возможных способов решения проблем, активизируя мышление студентов и раскрывая личностный потенциал каждого учащегося.
- Технология развития критического мышления – способствует формированию разносторонней личности, способной критически относиться к информации, умению отбирать информацию для решения поставленной задачи.

Комплексное использование в учебном процессе всех вышеназванных технологий стимулируют личностную, интеллектуальную активность, развивают познавательные процессы, способствуют формированию компетенций, которыми должен обладать будущий специалист.

Основные виды интерактивных образовательных технологий включают в себя:

- работа в малых группах (команде) - совместная деятельность студентов в группе под руководством лидера, направленная на решение общей задачи путём творческого сложения результатов индивидуальной работы членов команды с делением полномочий и ответственности;
- проектная технология - индивидуальная или коллективная деятельность по отбору, распределению и систематизации материала по определенной теме, в результате которой составляется проект;
- анализ конкретных ситуаций - анализ реальных проблемных ситуаций, имевших место в соответствующей области профессиональной деятельности, и поиск вариантов лучших решений;
- развитие критического мышления – образовательная деятельность, направленная на развитие у студентов разумного, рефлексивного мышления, способного выдвинуть новые идеи и увидеть новые возможности.

Подход разбора конкретных задач и ситуаций широко используется как преподавателем, так и студентами во время лекций, лабораторных занятий и анализа результатов самостоятельной работы. Это обусловлено тем, что при исследовании и решении каждой конкретной задачи имеется, как правило, несколько методов, а это требует разбора и оценки целой совокупности конкретных ситуаций.

Темы, задания и вопросы для самостоятельной работы призваны сформировать навыки поиска информации, умения самостоятельно расширять и углублять знания, полученные в ходе лекционных и практических занятий.

Подход разбора конкретных ситуаций широко используется как преподавателем, так и студентами при проведении анализа результатов самостоятельной работы.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,
– в форме электронного документа.
Для лиц с нарушениями слуха:
– в печатной форме,
– в форме электронного документа.
Для лиц с нарушениями опорно-двигательного аппарата:
– в печатной форме,
– в форме электронного документа.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4. Оценочные и методические материалы

4.1 Оценочные средства для текущего контроля успеваемости и промежуточной аттестации

Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «Объектно-ориентированное программирование».

Оценочные средства включает контрольные материалы для проведения **текущего контроля** в форме оценки лабораторных работ к проекта к **экзамену**.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине (модулю) предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:
– в печатной форме увеличенным шрифтом,
– в форме электронного документа.
Для лиц с нарушениями слуха:
– в печатной форме,
– в форме электронного документа.
Для лиц с нарушениями опорно-двигательного аппарата:
– в печатной форме,
– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Структура оценочных средств для текущей и промежуточной аттестации

			Наименование

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или ее части)	оценочного средства	
			Текущий контроль	Промежуточная аттестация
1.	Введение в платформу Java и среду разработки	PL-2	*ЛР 1*	Вопросы к экзамену 1
2.	Базовые конструкции языка	ОПК-3 PL-2	*ЛР 2*	Вопросы к экзамену 2 - 7
3.	Классы и объекты. Инкапсуляция	ОПК-3 PL-2	*ЛР 3-4*	Вопросы к экзамену 8 - 10
4.	Наследование и полиморфизм	ОПК-3 PL-2	*ЛР 5*	Вопросы к экзамену 11 - 14
5.	Абстрактные классы и интерфейсы	ОПК-3 PL-2	*ЛР 6*	Вопросы к экзамену 15 - 17
6.	Исключения	ОПК-3 PL-2	*ЛР 7*	Вопросы к экзамену 18-19
7.	Коллекции и обобщения (Generics)	ОПК-3 PL-2	*ЛР 8*	Вопросы к экзамену20-22
8.	Основы юнит-тестирования (JUnit)	ОПК-3 PL-2	*ЛР 9*	Вопросы к экзамену 23-24
9.	ИИ-инструменты в разработке: генерация кода и тестов	ОПК-3 PL-2	*ЛР 10*	Вопросы к экзамену 25-30

Показатели, критерии и шкала оценки сформированных компетенций

Соответствие **базовому уровню** освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **удовлетворительно**):

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 Аргументировано применяет современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения
Знает основные конструкции объектно-ориентированного языка JAVA, необходимые для создания программных продуктов и программных комплексов различного назначения, в т.ч. для анализа данных
Умеет разрабатывать алгоритмы и объектно-ориентированные программы для анализа данных на языке JAVA, готовые к интеграции в реальные системы
Владеет современными информационными технологиями, в том числе базовыми методами объектно-ориентированного программирования на платформе IntelliJ IDEA

ОПК-3.2 Ориентируется в современных положениях и концепциях прикладного и системного программного обеспечения, архитектуры компьютеров и сетей (в том числе и глобальных), технологии создания и сопровождения программных продуктов и программных комплексов

Знает концепции, положенные в основу системного программного обеспечения IntelliJ IDEA
Умеет применять математические методы для построения экономически эффективных алгоритмов с учетом архитектуры компьютеров
Владеет основами формализации задач для объектно-ориентированного проектирования при разработке эффективных программных решений, создании программных продуктов

PL-2 ***Способен применять JVM-совместимые языки программирования для решения задач в области ИИ***

PL-2.1 *Разрабатывает и отлаживает прикладные решения разного уровня сложности и для широкого круга конечных пользователей с использованием JVM-совместимых языков программирования, тестирует, испытывает и оценивает качество таких решений*
Знает:
Синтаксис Java и базовые конструкции ООП (инкапсуляция, наследование, полиморфизм).
Назначение интерфейсов, абстрактных классов, коллекций, generics и исключений.
Принципы применения YandexGPT/GigaChat для генерации кода и тестов JUnit..
Умеет:
Проектировать иерархии классов и писать модульные тесты в IntelliJ IDEA.
Формулировать промпты для ИИ-ассистентов на получение Java-кода и тестов.
Критически проверять и дорабатывать предложенный ИИ код до рабочего состояния.
Владеет навыками:
Полного цикла «запрос к ИИ → валидация → интеграция» при решении учебных задач.
Составления точных запросов, обеспечивающих воспроизводимый результат генерации.
Ускоренного прототипирования с обязательным осмысленным контролем каждого сгенерированного фрагмента.

Соответствие **продвинутому уровню** освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **хорошо**):

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 Аргументировано применяет современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения
Знает синтаксис и семантику объектно-ориентированного языка JAVA, используемого для создания программных продуктов и программных комплексов различного назначения, в т.ч. для анализа данных
Умеет разрабатывать алгоритмы и объектно-ориентированные программы для анализа данных на языке JAVA, соответствующие стандартам качества и готовые к интеграции в реальные системы
Владеет современными информационными технологиями при создании программных продуктов и программных комплексов различного назначения, в

том числе базовыми методами объектно-ориентированного программирования на платформе IntelliJ IDEA

ОПК-3.2 Ориентируется в современных положениях и концепциях прикладного и системного программного обеспечения, архитектуры компьютеров и сетей (в том числе и глобальных), технологии создания и сопровождения программных продуктов и программных комплексов
Знает концепции, в т.ч. экономические, положенные в основу системного программного обеспечения IntelliJ IDEA
Умеет применять математические методы для построения экономически эффективных алгоритмов с учетом архитектуры компьютеров и сетей
Владеет основами формализации задач для объектно-ориентированного проектирования при разработке эффективных программных решений, создании и сопровождении программных продуктов и программных комплексов

PL-2 **Способен применять JVM-совместимые языки программирования для решения задач в области ИИ**

PL-2.1 ***Разрабатывает и отлаживает прикладные решения разного уровня сложности и для широкого круга конечных пользователей с использованием JVM-совместимых языков программирования, тестирует, испытывает и оценивает качество таких решений***
Знает:
Синтаксис Java и базовые конструкции ООП (инкапсуляция, наследование, полиморфизм).
Назначение интерфейсов, абстрактных классов, коллекций, generics и исключений.
Принципы применения YandexGPT/GigaChat для генерации кода и тестов JUnit..
Умеет:
Качественно проектировать иерархии классов и писать модульные тесты в IntelliJ IDEA.
Формулировать промпты для ИИ-ассистентов на получение Java-кода и тестов.
Критически проверять и дорабатывать предложенный ИИ код до рабочего состояния.
Владеет навыками:
Полного цикла «запрос к ИИ → валидация → интеграция» при решении учебных задач.
Составления точных запросов, обеспечивающих воспроизводимый результат генерации.
Ускоренного качественного прототипирования с обязательным осмысленным контролем каждого сгенерированного фрагмента.

Соответствие **экспертному уровню** освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **отлично**):

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 Аргументировано применяет современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения

Знает синтаксис, семантику и прагматику объектно-ориентированного языка JAVA, используемого для создания программных продуктов и программных комплексов различного назначения, в т.ч. для анализа данных и решения задач искусственного интеллекта

Умеет разрабатывать алгоритмы и объектно-ориентированные программы для анализа данных на языке JAVA, соответствующие стандартам качества и готовые к интеграции в реальные системы; оценивать сложность алгоритмов

Владеет современными информационными технологиями при создании программных продуктов и программных комплексов различного назначения, в том числе методами объектно-ориентированного программирования на платформе IntelliJ IDEA; методами разработки модульных и масштабируемых программ

ОПК-3.2 Ориентируется в современных положениях и концепциях прикладного и системного программного обеспечения, архитектуры компьютеров и сетей (в том числе и глобальных), технологии создания и сопровождения программных продуктов и программных комплексов

Знает концепции, в т.ч. экономические, положенные в основу системного программного обеспечения IntelliJ IDEA, а также концепцию паттернов объектно-ориентированного программирования

Умеет применять математические и вычислительные методы для построения экономически эффективных алгоритмов и структур данных с учетом архитектуры компьютеров и сетей различных масштабов

Владеет основами формализации прикладных и научных задач для объектно-ориентированного проектирования при разработке эффективных программных решений, создании и сопровождении программных продуктов и программных комплексов различного назначения

PL-2 ***Способен применять JVM-совместимые языки программирования для решения задач в области ИИ***

PL-2.1 *Разрабатывает и отлаживает прикладные решения разного уровня сложности и для широкого круга конечных пользователей с использованием JVM-совместимых языков программирования, тестирует, испытывает и оценивает качество таких решений*

Знает:

Синтаксис Java и базовые конструкции ООП (инкапсуляция, наследование, полиморфизм).

Назначение интерфейсов, абстрактных классов, коллекций, generics и исключений.

Принципы применения YandexGPT/GigaChat для генерации кода и тестов JUnit..

Умеет:

Качественно проектировать иерархии классов и писать модульные тесты в IntelliJ IDEA.

Формулировать промпты для ИИ-ассистентов на получение Java-кода и тестов на высоком уровне.

Критически проверять и дорабатывать предложенный ИИ код до рабочего состояния.

Владеет навыками:

Полного цикла «запрос к ИИ → валидация → интеграция» при решении учебных задач на высоком уровне.
Составления точных запросов, обеспечивающих воспроизводимый результат генерации.
Ускоренного качественного прототипирования с обязательным осмысленным контролем каждого сгенерированного фрагмента на высоком уровне.

Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Примерные темы лабораторных работ

Лабораторная работа 1. Установка инструментов. Hello World (2 ч)
1. «Установка JDK и IntelliJ IDEA.» Скачать и настроить JDK 17+, IDE, создать проект; проверить работоспособность на программе, выводящей информацию о системе («System.getProperty»).
2. «Первый запуск и отладка.» Написать программу «Анкета»: запрашивает имя и возраст, выводит приветствие. Освоить точки останова и пошаговое выполнение.

Лабораторная работа 2. Управляющие конструкции и массивы (2 ч)
3. «Калькулятор с консольным меню.» Реализовать простой калькулятор (+, -, , /) с выбором операции через «switch» и защитой от деления на ноль.
4. «Обработка двумерного массива.» Сгенерировать матрицу случайных чисел, найти максимальный элемент, сумму элементов главной диагонали, вывести матрицу в отформатированном виде.

Лабораторная работа 3. Классы и объекты. Инкапсуляция (4 ч)
5. «Класс «Студент».» Поля: ФИО, курс, средний балл. Конструкторы (по умолчанию, с параметрами), геттеры/сеттеры с проверкой (балл от 0 до 5). Метод «toString()» и демонстрация.
6. «Класс «Время».» Хранение часов, минут, секунд. Методы сложения двух времен, вывода в формате «чч:мм:сс». Полная инкапсуляция с валидацией значений.

Лабораторная работа 4. Наследование и полиморфизм (4 ч)
7. «Геометрические фигуры.» Базовый класс «Shape» с методом «getArea()». Наследники «Circle» (радиус) и «Rectangle» (стороны). Полиморфный вывод площадей массива фигур.
8. «Транспортные средства.» Класс «Vehicle» (скорость, вместимость). Наследники «Car» и «Bicycle». Переопределение метода «move()». Хранение в массиве и вызов для каждого.

Лабораторная работа 5. Абстрактные классы и интерфейсы (4 ч)
9. «Платежные системы.» Интерфейс «Payable» с методом «pay(double amount)». Реализации «CardPayment», «CashPayment». Добавить абстрактный класс «Account» с полем баланс и методом «checkBalance()».
10. «Сортировка объектов.» Интерфейс «Comparable<Student>« для класса «Student» (по среднему баллу). Создать список студентов, отсортировать через «Collections.sort()».

Лабораторная работа 6. Исключения (2 ч)
11. «Банкомат с ограничениями.» Модифицировать «Account»: снятие наличных при недостатке средств вызывает собственное исключение «InsufficientFundsException».

Сгенерировать через YandexGPT код кастомного исключения и блок обработки, вставить и проверить.
12. «Форма регистрации.» Программа проверяет корректность введённого email и возраста. При ошибках выбрасываются «InvalidEmailException» и «InvalidAgeException». ИИ генерирует классы исключений и метод валидации, студент дорабатывает и покрывает тестами.

Лабораторная работа 7. Генерация кода и модульных тестов с ИИ (4 ч)
13. «Генерация модели «Библиотека» По промпту «Создай классы Book, Library с методами добавления, поиска по автору и выдачи книг» получить от YandexGPT код. Проверить, дополнить инкапсуляцией, сгенерировать JUnit-тесты через GigaChat и добиться их прохождения.
14. «Генерация CRUD-репозитория.» Получить от ИИ реализацию интерфейса «Repository<T>« с методами «add», «remove», «findById» на основе «Map». Самостоятельно написать юнит-тесты (или запросить у ИИ и отладить).

Лабораторная работа 8. Коллекции и Generics (4 ч)
15. «Контейнер с параметризованным поиском.» Реализовать обобщённый класс «Box<T>«, хранящий список элементов. Методы фильтрации по предикату, поиска максимума (если «T extends Comparable»). Методы фильтрации и сортировки запросить у GigaChat, разобрать и скорректировать.
16. «Статистика текста.» Для заданной строки, с помощью «Map<String, Integer>« подсчитать частоту слов. ИИ генерирует метод приведения к нижнему регистру и удаления знаков препинания, студент встраивает и тестирует.

Лабораторная работа 9. Работа с файлами (ввод-вывод) (2 ч)
17. «Сохранение и загрузка списка студентов в JSON.» Используя библиотеку org.json (или встроенные средства), записать коллекцию объектов «Student» в файл и считать обратно. ИИ генерирует шаблон сериализации/десериализации, студент адаптирует под свой класс.
18. «Логирование действий.» Класс «Logger» записывает сообщения с меткой времени в текстовый файл. Промпт для GigaChat: «Создай метод appendToLog с синхронизацией и обработкой IOException».

Лабораторная работа 10. Итоговый мини-проект с обязательным применением ИИ (6 ч)
19. «Приложение «Менеджер задач».» Консольное приложение с возможностью добавлять, удалять, отмечать выполненные задачи, хранить их в файле. С помощью YandexGPT спроектировать архитектуру (классы Task, TaskManager, FileStorage), получить реализацию отдельных модулей и полный набор JUnit-тестов. Студент координирует сборку, исправляет ошибки, демонстрирует работающее приложение и прохождение тестов.
20. «Приложение «Заметки с категориями».» Заметка имеет заголовок, текст, категорию и дату. Реализовать сортировку по дате, фильтрацию по категории, сохранение/загрузку из JSON. Взаимодействие с ИИ: запрос на генерацию классов и тестов, валидация, защита проекта с пояснением всех внесённых в сгенерированный код правок.

Тематика заданий самостоятельной работы

1. Настройка JDK 17+, IntelliJ IDEA. Написание простейшей программы, знакомство с отладчиком.
2. Решение задач на условные операторы, циклы, обработку одномерных и двумерных массивов.
3. Создание класса «Студент», «Время» или «Банковский счёт» с полями, конструкторами, методами доступа.

4. Иерархия «Фигура» → «Круг», «Прямоугольник»; транспортные средства с переопределением метода `move()`.
5. Реализация платежных систем (интерфейс `Payable`) и сортировка объектов через `Comparable`.
6. Генерация собственных исключений (например, `InsufficientFundsException`). Использование `try-catch-finally`. Запрос шаблонов обработчиков у ИИ.
7. Разработать классы:
   - `Image` – хранит ширину, высоту, массив пикселей (можно заглушку).
   - `BoundingBox` – описывает прямоугольную область (координаты, ширина, высота).
   - `Detector` – содержит метод `List<BoundingBox> detect(Image image)`, который возвращает найденные объекты (имитируется простейшим алгоритмом, например, поиск областей с яркостью выше порога).
     Реализовать полиморфную замену детектора (интерфейс `ObjectDetector` с разными стратегиями).
     Написать JUnit-тесты, проверяющие корректность работы детектора на тестовых изображениях (созданных в коде).
8. Создание параметризованного класса `Box<T>` с методами фильтрации. Реализация репозитория на основе `Map`.

9. Написать программу для анализа тональности текстов, используя коллекции:
   - Получить от ИИ (YandexGPT/GigaChat) тональность (положительная, нейтральная, отрицательная) для нескольких предложений.
   - Результаты сохранить в `Map<String, Sentiment>`, где ключ – текст, значение – тональность.
   - Написать обобщённый метод `printGroupedBySentiment(Map<String, Sentiment> data)`, который группирует тексты по тональности и выводит статистику (использовать `Map<Sentiment, List<String>>`).
   - Продемонстрировать работу на 5–7 примерах, включая граничные случаи.
10. Сохранение и загрузка коллекции объектов (например, результатов детекции или анализируемых текстов) в JSON. Использование библиотеки `org.json` или `Jackson`. Генерация шаблонов через ИИ.
11. Студент выбирает одну из двух линий:
    - Детектор объектов на синтетических данных (расширение работы №7): добавить графический интерфейс для визуализации bounding boxes, реализовать загрузку реальных изображений (через BufferedImage), сохранять результаты в JSON.
    - Анализатор тональности с накоплением статистики (расширение работы №8): добавить хранение истории запросов, возможность экспорта отчёта в CSV, использовать коллекции и generics для обработки произвольного набора текстов.
      В проекте обязательно документировать все промпты, внесённые правки в сгенерированный код, а также предоставить полный набор проходящих JUnit-тестов.

Зачетно-экзаменационные материалы для промежуточной аттестации (экзамен)

Вопросы для подготовки к экзамену

1. Ниже приведены 30 вопросов по всем лекциям курса. Формулировки максимально сжаты.

1. Из каких компонентов состоит платформа Java? Что такое JDK, JRE и JVM?
2. Какие примитивные типы данных есть в Java? Чем отличается «int» от «Integer»?
3. Как выглядит точка входа в программу на Java? Что означает сигнатура «public static void main(String[] args)»?
4. Чем отличаются операторы «==« и «equals()» при сравнении объектов?

5. В каких случаях «switch»-конструкция может заменить цепочку «if-else»? Приведите синтаксис.

6. Чем отличаются циклы «for», «while» и «do-while»? Когда какой уместнее?

7. Как объявить двумерный массив? Как обратиться к его элементу?

8. Что такое класс и объект? Как создать экземпляр класса через оператор «new»?

9. Какие модификаторы доступа вы знаете? Поясните разницу между «private», «default», «protected», «public».

10. Что такое инкапсуляция? Как она реализуется с помощью геттеров и сеттеров?

11. Что такое перегрузка методов? Чем она отличается от переопределения?

12. Как работает ключевое слово «extends»? Что наследуется от суперкласса, а что нет?

13. Что делает ключевое слово «super»? Приведите примеры использования в конструкторе и методе.

14. Что такое полиморфизм и динамическое связывание? Как тип ссылки влияет на вызываемый метод?

15. Зачем нужны абстрактные классы? Можно ли создать их экземпляр?

16. Чем интерфейс отличается от абстрактного класса? Можно ли реализовать несколько интерфейсов?

17. Что такое функциональный интерфейс? Сколько в нём может быть абстрактных методов?

18. Как устроен блок «try-catch-finally»? В каком порядке выполняются секции при возникновении исключения?

19. Для чего используется «throws» в сигнатуре метода? Как создать собственное исключение?

20. Какие основные интерфейсы коллекций вы знаете? Приведите примеры реализаций для «List», «Set», «Map».

21. Что такое generics? Как объявить обобщённый класс с параметром типа «<T>«?

22. Как работает wildcard («?») в generics? Чем отличаются «<? extends T>« и «<? super T>«?

23. Какие аннотации JUnit 5 обязательны для модульного теста? Что делает «@Test»?

24. Какие методы-утверждения «assertEquals» и «assertThrows» вы знаете? Для чего нужен «@BeforeEach»?

25. Какие отечественные ИИ-сервисы могут использоваться для генерации кода на Java? Назовите два.

26. Из чего состоит хороший промпт для YandexGPT при запросе Java-кода? Приведите структуру.

27. Какие типовые ошибки допускает ИИ при генерации ООП-кода? Как их выявить?

28. Какие этические нормы следует соблюдать, используя сгенерированный ИИ код в учебных работах?

29. Как связаны принципы ООП и модульное тестирование? Почему инкапсуляция упрощает написание тестов?

30. Опишите полный цикл разработки мини-проекта с применением ИИ: от промпта до сдачи работающего приложения с тестами.

4.2 Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Методические рекомендации, определяющие процедуры оценивания лабораторных работ:

В дисциплине «Объектно-ориентированное программирование» лабораторные работы являются основной формой текущего контроля успеваемости и инструментом формирования заявленных компетенций (ОПК-2, ОПК-6, ОПК-7, PL-2). Всего студент выполняет 10 лабораторных работ, каждая из которых завершается оформлением письменного отчёта и устной защитой. Итоговая оценка за дисциплину складывается из результатов текущего контроля (лабораторные работы) и ответа на экзамене, причём лабораторные работы составляют весомую часть подготовки к экзамену.

Принципы оценивания

Оценивание каждой лабораторной работы строится на трёх ключевых принципах:

1. Практическая работоспособность – код должен компилироваться, запускаться и выполнять заданную функцию без критических ошибок.
2. Соответствие принципам ООП – даже на ранних работах (управляющие конструкции, массивы) поощряется аккуратное оформление, а начиная с работы №3 обязательно применение инкапсуляции, наследования, полиморфизма и интерфейсов там, где это уместно.
3. Документирование взаимодействия с ИИ – в работах №6–10 студент обязан фиксировать промпты, отправленные в YandexGPT или GigaChat, анализировать сгенерированный код и указывать, какие правки были внесены для достижения рабочего состояния. Без этого лабораторная работа не засчитывается.

Для каждой лабораторной работы преподаватель использует единый набор критериев, адаптированный под сложность и тему работы. Базовый чек-лист включает:

- Полнота выполнения задания (0–2 балла) – реализованы ли все требуемые методы, классы, функции.
- Корректность и стиль кода (0–2 балла) – имена переменных, отсутствие дублирования, разумная длина методов, комментарии к сложным фрагментам.
- Применение ООП (0–2 балла) – для работ №3–10: инкапсуляция, наследование/полиморфизм (где нужно), использование абстрактных классов/интерфейсов. Для работ №1–2 этот критерий не применяется или заменяется на «логическую правильность алгоритмов».
- Наличие и прохождение тестов (0–2 балла) – начиная с работы №6 (исключения, ИИ) требуются элементарные JUnit-тесты; для работ №7–10 тесты обязательны и должны быть зелёными.
- Использование ИИ и рефлексия (0–2 балла) – только для работ №6–10: приведены ли промпты, описан ли процесс валидации и доработки кода.
- Защита (устные ответы) (0–2 балла) – студент должен объяснить, как работает программа, почему выбрана та или иная структура, какие трудности возникли при интеграции ИИ-кода.

Каждый критерий оценивается по шкале: 0 – не выполнено / неудовлетворительно, 1 – выполнено частично или с замечаниями, 2 – выполнено полностью и качественно. Максимальный балл за одну лабораторную работу – 12. Для получения зачёта по работе необходимо набрать не менее 7 баллов, при этом обязательно иметь хотя бы 1 балл по критерию «работоспособность» и (для работ с ИИ) по критерию «использование ИИ».

Специфика оценивания по этапам курса

Первые работы (№1–2: установка инструментов, управляющие конструкции, массивы)

Здесь основной упор делается на корректность синтаксиса, умение пользоваться отладчиком и средой IntelliJ IDEA. Оценивается также аккуратность оформления отчёта. Критерий ООП пока не применяется.

Работы по основам ООП (№3–5: классы, инкапсуляция, наследование, полиморфизм, абстрактные классы и интерфейсы)

Помимо работоспособности, строго проверяется сокрытие данных (private поля,

геттеры/сеттеры), правильное использование extends и implements, отсутствие «божественных классов». Если студент использует ИИ для генерации этих работ, он обязан указать это в отчёте – такие работы оцениваются по тем же критериям, но с дополнительным вопросом на защите: «Какие принципы ООП нарушил сгенерированный код и как вы их исправили?».

Работы с исключениями, коллекциями, тестами (№6–9)
Здесь впервые появляется требование наличия JUnit-тестов (минимум 3–5 утверждений на ключевой метод). Оценивается не только сам код, но и умение студента генерировать тесты через ИИ, а затем адаптировать их под свою предметную область. При проверке преподаватель может запустить тесты и убедиться, что они зелёные. Если тесты отсутствуют или падают, работа отправляется на доработку.

Итоговый мини-проект (работа №10)
Оценивается по отдельной, более детальной шкале (см. «Критерии оценки итогового мини-проекта»), которая включает функциональную полноту, качество ООП, покрытие тестами (≥80%), обязательное документирование всех промптов и правок, стабильность работы, сериализацию в JSON и устную защиту. Максимальный балл за итоговый проект – 100, который затем переводится в оценку «отлично» (86–100), «хорошо» (70–85), «удовлетворительно» (50–69) или «неудовлетворительно» (<50). При этом проекты, выполненные без использования ИИ, автоматически получают «неудовлетворительно» независимо от качества кода.

В лабораторных работах №6–10 студенты активно используют YandexGPT или GigaChat для получения Java-кода и тестов. Чтобы оценить качество такого кода, в дисциплине применяются шесть основных метрик, каждая из которых по-своему привязана к конкретным заданиям.

Компилируемость проверяется сразу после вставки сгенерированного фрагмента: код не должен содержать синтаксических ошибок, отсутствующих импортов или несовместимости типов. Этот критерий важен для всех работ, начиная с первой, но особенно строго – в №6 (исключения), №7 (генерация модели «Библиотека») и в итоговом проекте №10, где компилируемость должна достигаться с первой-второй попытки.

Проходимость тестов – ключевая метрика для работ №7–10. Студент обязан добиться, чтобы сгенерированные или им доработанные JUnit-тесты стали зелёными. Для ЛР №7 и №10 требуется 100% прохождения, для ЛР №8 и №9 – не менее 80%. Падающие тесты возвращаются на доработку.

Соответствие стилю кодирования оценивается начиная с ЛР №3 (классы и инкапсуляция) и до итогового проекта. Проверяется именование переменных, наличие Javadoc, разумная длина методов, отсутствие магических чисел. Если студент получил от ИИ «грязный» код, он обязан привести его в порядок; систематические нарушения стиля ведут к снижению оценки.

Семантическая корректность и полнота по ТЗ – метрика того, насколько сгенерированное решение действительно решает поставленную задачу, включая граничные случаи (пустые коллекции, null, некорректный ввод). Особенно важна для ЛР №6 (исключения в нештатных ситуациях), №8 (фильтрация и поиск с предикатами) и №10 (полный функционал приложения).

ООП-качество проверяет, исправил ли студент типичные процедурные ошибки ИИ: публичные поля, отсутствие инкапсуляции, неоправданное использование static. Эта метрика вводится с ЛР №3 и становится обязательной в работах с ИИ (№6–10). В отчёте студент должен указать, какие принципы ООП нарушал сгенерированный код и как он их восстановил.

Документирование использования ИИ – уникальная метрика курса. Начиная с ЛР №6 и до итогового проекта студент фиксирует точные промпты, полученный код, список своих правок и выводы о типичных ошибках модели. Без такого отчёта работа не засчитывается.

Таким образом, каждая лабораторная работа оценивается по набору метрик, адаптированных к её сложности: ранние работы (№1–2) проверяют лишь компилируемость, стиль и семантику; работы №3–5 добавляют ООП-качество; а с №6 по №10 включаются также тесты и обязательная рефлексия над кодом, сгенерированным искусственным интеллектом. Такой подход учит студентов не слепо доверять ИИ, а критически анализировать и улучшать его результаты.

Все лабораторные работы в совокупности формируют портфолио, которое студент предъявляет на экзамене. Экзаменатор вправе задать уточняющие вопросы по любой из лабораторных работ, и качество ответов влияет на итоговую экзаменационную оценку. Таким образом, систематическое выполнение и защита лабораторных работ с соблюдением описанных критериев является залогом успешного освоения дисциплины и получения высокой оценки.

Методические рекомендации, определяющие процедуры оценивания самостоятельной работы:

Вид СРС	**Форма контроля**	**Периодичность**	**Фиксация результата**
Изучение теоретического материала (лекции, учебники, документация Java)	Устный опрос на лабораторном занятии, тестирование в Moodle, контрольные вопросы при защите ЛР	Перед каждой лабораторной работой	Оценка в журнале текущей успеваемости
Решение задач на управляющие конструкции, массивы, основы ООП	Проверка домашних заданий (код + скриншоты вывода), мини-тесты в начале занятия	К занятиям №2–5	Отметка о выполнении / зачтено
Выполнение индивидуальных заданий (доработка кода, полученного от ИИ)	Защита отчёта по лабораторной работе (включая анализ промптов и правок)	По каждой ЛР №6–10	Оценка в составе лабораторной работы
Подготовка к экзамену (повторение, проработка вопросов)	Промежуточная аттестация (экзамен)	В конце семестра	Экзаменационная оценка

Критерии оценки самостоятельной работы
Оценивание производится по 100-балльной шкале с последующим переводом в оценку (отлично, хорошо, удовлетворительно, неудовлетворительно). Для разных форм СРС применяются специализированные критерии.
Оценка теоретической подготовки (изучение материала)

- **Отлично (86-100 баллов)** – студент свободно отвечает на контрольные вопросы, демонстрирует понимание не только базовых, но и дополнительных концепций (например, отличие полиморфизма от перегрузки, принципы работы JVM).
- **Хорошо (70-85)** – твёрдое знание основных понятий, допускаются незначительные неточности (например, путаница между абстрактным классом и интерфейсом), но после наводящего вопроса ответ исправляется.
- **Удовлетворительно (50-69)** – базовые знания присутствуют, но ответы фрагментарны, требуется много дополнительных вопросов.
- **Неудовлетворительно (<50)** – студент не может ответить на простейшие вопросы (например, назвать принципы ООП или синтаксис цикла for).

Оценка выполнения домашних заданий (код без ИИ или с ИИ)
Для заданий, не входящих в состав лабораторных работ (например, задачи на массивы или простые классы), применяются критерии:

Критерий	Вес	Описание
Полнота решения	40%	Задача решена полностью, все требования выполнены
Корректность	30%	Код работает без ошибок, граничные случаи обработаны
Стиль и оформление	15%	Именование, отступы, комментарии, отсутствие копипасты
Самостоятельность	15%	Работа выполнена без списывания, студент может объяснить код

Итоговая оценка за домашнее задание = сумма баллов по критериям, переведённая в шкалу 0-100.
Оценка работы с ИИ (документирование промптов и правок)
Для заданий, где студент использует YandexGPT/GigaChat (ЛР №6-10), самостоятельная работа оценивается в составе отчёта по лабораторной работе. Ключевые элементы:

- **Качество промптов** (0-10 баллов) – насколько точно описан контекст, указаны типы данных, примеры входов/выходов, требования к стилю.
- **Полнота полученного ответа** (0-10) – ИИ выдал рабочий фрагмент, который требует минимальных правок, или код совершенно не подходит.
- **Анализ ошибок и доработка** (0-20) – студент перечислил, какие ошибки допустил ИИ (нарушение инкапсуляции, отсутствие обработки исключений, неэффективные алгоритмы) и как их исправил.
- **Рефлексия** (0-10) – выводы о применимости ИИ для данной задачи, предложения по улучшению промптов.

Максимум 50 баллов за эту часть, которые затем суммируются с баллами за работоспособность кода и тесты.
Процедура оценивания СРС

1. **Регулярность** – самостоятельная работа проверяется не реже одного раза в две недели (к моменту очередной лабораторной работы).

2. **Формат сдачи** – все домашние задания и отчёты загружаются в систему Moodle или в Git-репозиторий студента (ссылка предоставляется преподавателю).
3. **Сроки** – каждая работа должна быть сдана не позднее, чем за 3 дня до защиты соответствующей лабораторной работы. Просрочка без уважительной причины снижает максимальную оценку на 20%.
4. **Пересдача** – при получении оценки «неудовлетворительно» студент имеет право доработать задание в течение 1 недели и сдать повторно. Максимальная оценка за пересдачу – 70 баллов.
5. **Интеграция с лабораторными работами** – многие виды СРС являются подготовительным этапом к лабораторным работам. Качество самостоятельной подготовки напрямую влияет на успешность защиты ЛР. Преподаватель может отказать в допуске к лабораторной работе, если студент не выполнил обязательное домашнее задание.

Методические рекомендации, определяющие процедуры оценивания на экзамене:

Процедура промежуточной аттестации проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

Итоговой формой контроля сформированности компетенций у обучающихся по дисциплине является зачет. Студенты обязаны сдать зачет в соответствии с расписанием и учебным планом.

ФОС промежуточной аттестации состоит из заданий и результатов текущего контроля.

Форма проведения экзамена: устно.

Преподавателю предоставляется право задавать студентам дополнительные вопросы по всей учебной программе дисциплины.

Результат сдачи экзамена заносится преподавателем в экзаменационную ведомость и зачетную книжку.

Оценивание уровня освоения дисциплины основывается на качестве выполнения студентом заданий текущего контроля и ответов на вопросы экзамена.

Критерии оценки:

оценка «неудовлетворительно» выставляется в случае выполнения одного из условий:

- непонимание сущности излагаемых вопросов, грубые ошибки в ответе, неуверенные и неточные ответы на дополнительные вопросы;
- выполнено менее 50% контрольных заданий.

оценка «удовлетворительно» в случае выполнения условий:

- частично ответил на два вопроса экзаменационного билета или достаточно полно ответил хотя бы на один вопрос;
- выполнено не менее 50% контрольных заданий.

оценка «хорошо» в случае выполнения условий:

- достаточно полно ответил на два вопроса экзаменационного билета;
- даны частичные ответы на дополнительные вопросы;
- выполнено не менее 60% контрольных заданий.

оценка «отлично» в случае выполнения условий:

- глубокие исчерпывающие знания по вопросам экзаменационного билета;
- даны правильные и конкретные ответы на дополнительные вопросы;
- сданы все тесты и выполнено не менее 80% контрольных заданий.

Методические рекомендации, определяющие процедуры оценивания лабораторных работ:

Процедура оценивания лабораторных работ проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

По каждой лабораторной работе оформляется отчет. Отчеты сдаются на проверку руководителю в течение курса по мере их выполнения, и защищаются студентами в установленном порядке.

При защите отчета студенту могут быть заданы вопросы и дополнительные задания по сути лабораторной работы, в том числе из списка контрольных вопросов к данной лабораторной работе. При неудовлетворительной оценке знаний студента по теме данного отчета, студент возвращается к повторному изучению соответствующих материалов, после чего допускается к повторной защите. Неудовлетворительно выполненный отчет также возвращается на доработку.

Отчет должен содержать заголовок, тему лабораторной работы, цель, задание, индивидуальную тему, описание хода выполнения работы, необходимые прикладные материалы (схемы, макеты документов и т.п.), в соответствии с требованиями к содержанию, и выводы по работе.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4.3. Методические указания по организации вычислительной инфраструктуры

Условия применения:

- Курс рассчитан на студентов 2 года обучения.
- Наличие доступа к вычислительным ресурсам (GitLab, компиляторы Java).
- Индивидуальные задания включают задачи на реализацию алгоритмов и структур данных, проверяемые автотестами.

Цели, задачи и ожидаемые результаты:

Цели:

– Обеспечить студентов инструментами для работы с алгоритмами и структурами данных.

– Приучить к использованию Git, CI/CD и автотестов для контроля качества кода.

Задачи преподавателя:

1. Создание учетных записей студентов в GitLab.
2. Настройка GitLab Runner для автоматического тестирования.
3. Разработка шаблонного репозитория для лабораторных работ.
4. Написание автотестов для проверки корректности реализации алгоритмов.
5. Визуализация результатов тестирования.

Ожидаемые результаты студентов:

– Умение работать с Git и CI/CD.

– Навыки написания чистого, тестируемого кода на Java.

– Понимание важности автоматической проверки алгоритмов.

Порядок реализации:

1. **Создание учетных записей в GitLab:**

Каждый студент получает доступ к приватному репозиторию.

2. **Настройка GitLab Runner:**

Используется Docker-образ с компилятором Java и фреймворками для тестирования.

3. **Шаблонный репозиторий:**

Включает:

– `.gitlab-ci.yml` для автоматического тестирования.

– `README.md` с инструкциями.

– Примеры кода и тестов.

4. **Автотесты:**

Проверяют корректность реализации алгоритмов (например, сортировки, работы с деревьями, хеш-таблицами).

5. **Визуализация результатов:**

Генерация отчетов с указанием успешных и проваленных тестов.

Порядок проверки корректности:

Чек-лист:

– Наличие Git-репозиториев у всех студентов.

– Корректная работа CI/CD.

– Автотесты покрывают ключевые функции.

4.4. Методические указания по организации лабораторных работ

Условия применения:

– Курс включает 17 лабораторных работ (см. раздел 2.3.3 РПД).

– Для выполнения требуется:

Среда разработки (IntelliJ IDEA).

GitLab для сдачи заданий.

Цели, задачи и ожидаемые результаты:

Цели:

– Практическое освоение алгоритмов и структур данных.

– Развитие навыков отладки и оптимизации кода.

Задачи преподавателя:

1. Подготовка плана лабораторных работ.
2. Разработка индивидуальных заданий.

3. Организация Git-инфраструктуры и автотестов.

Ожидаемые результаты студентов:

- Умение реализовывать и анализировать алгоритмы.
- Навыки работы с Git и JUnit.

Порядок реализации:

1. **План лабораторных работ:**

Соответствует темам из РПД (сортировки, деревья, графы, хеш-таблицы и т. д.).

2. **Критерии оценки:**

- **зачтено:** Полная реализация, анализ сложности, код проходит тесты.
- **Незачтено:** Код не работает или не соответствует ТЗ.

Порядок проверки корректности:

Чек-лист:

- Наличие автотестов для каждой лабораторной работы.
- Инструкции по именованию коммитов и методов.

4.5. Методические указания по организации проектной деятельности студентов

Условия применения:

- Курс включает проектную работу (16 часов).
- Доступны кейсы от преподавателей.

Цели, задачи и ожидаемые результаты:

Цели:

- Применение алгоритмов и структур данных в реальных задачах.
- Развитие навыков командной работы.

Задачи преподавателя:

1. Сбор кейсов (например, оптимизация поиска в больших данных).
2. Формирование ТЗ для проектов.
3. Разработка системы оценки.

Ожидаемые результаты студентов:

Умение решать прикладные задачи с использованием изученных алгоритмов.

Порядок реализации:

1. **Пример кейса:**

Оптимизация хеш-таблицы для ускорения поиска в базе данных.

2. **ТЗ для проекта:**

- Реализовать хеш-таблицу с методами цепочек и открытой адресации.
- Сравнить производительность на реальных данных.

3. **Критерии оценки:**

- Корректность реализации.
- Анализ эффективности (O-нотация).

Порядок проверки корректности:

Чек-лист:

- Наличие минимум 3 кейса.
- Четкие критерии оценки проектов.

5. Перечень учебной литературы, информационных ресурсов и технологий

5.1 Учебная литература:

1. Тузовский, А. Ф. Объектно-ориентированное программирование : учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2021. — 206 с. — (Высшее образование). — ISBN 978-5-534-00849-4. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/470223

2. Бабушкина, И. А. Практикум по объектно-ориентированному программированию : [16+] / И. А. Бабушкина, С. М. Окулов. – 5-е изд., электрон. – Москва : Лаборатория знаний, 2020. – 369 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=221691 (дата обращения: 15.08.2021). – Библиогр.: с. 358. – ISBN 978-5-00101-780-6. – Текст : электронный.

– Документация GitLab.

5.2. Периодические издания:

– Базы данных компании «Ист Вью» http://dlib.eastview.com
– Электронная библиотека GREBENNIKON.RU https://grebennikon.ru/

5.3. Интернет-ресурсы, в том числе современные профессиональные базы данных и информационные справочные системы

Электронно-библиотечные системы (ЭБС):

1. ЭБС «ЮРАЙТ» https://urait.ru/
2. ЭБС «УНИВЕРСИТЕТСКАЯ БИБЛИОТЕКА ОНЛАЙН» http://www.biblioclub.ru/
3. ЭБС «BOOK.ru» https://www.book.ru
4. ЭБС «ZNANIUM.COM» www.znanium.com
5. ЭБС «ЛАНЬ» https://e.lanbook.com

Профессиональные базы данных

1. Scopus http://www.scopus.com/
2. ScienceDirect https://www.sciencedirect.com/
3. Журналы издательства Wiley https://onlinelibrary.wiley.com/
4. Научная электронная библиотека (НЭБ) http://www.elibrary.ru/
5. Полнотекстовые архивы ведущих западных научных журналов на Российской платформе научных журналов НЭИКОН http://archive.neicon.ru
6. Национальная электронная библиотека (доступ к Электронной библиотеке диссертаций Российской государственной библиотеки (РГБ) https://rusneb.ru/
7. Президентская библиотека им. Б.Н. Ельцина https://www.prlib.ru/
8. База данных CSD Кембриджского центра кристаллографических данных (CCDC) https://www.ccdc.cam.ac.uk/structures/
9. Springer Journals: https://link.springer.com/
10. Springer Journals Archive: https://link.springer.com/
11. Nature Journals: https://www.nature.com/
12. **Springer Nature Protocols and Methods**: https://experiments.springernature.com/sources/springer-protocols
13. Springer Materials: http://materials.springer.com/
14. Nano Database: https://nano.nature.com/
15. Springer eBooks (i.e. 2020 eBook collections): https://link.springer.com/
16. "Лекториум ТВ" http://www.lektorium.tv/
17. Университетская информационная система РОССИЯ http://uisrussia.msu.ru

Бесплатные образовательные ресурсы

1. Visual Studio Code – редактор кода
2. Google Scholar/arXiv – доступ к научным публикациям

Ресурсы свободного доступа

1. КиберЛенинка http://cyberleninka.ru/;
2. Американская патентная база данных http://www.uspto.gov/patft/
3. Министерство науки и высшего образования Российской Федерации https://www.minobrnauki.gov.ru/;
4. Федеральный портал "Российское образование" http://www.edu.ru/;
5. Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru/;
6. Единая коллекция цифровых образовательных ресурсов http://school-collection.edu.ru/ .
7. Проект Государственного института русского языка имени А.С. Пушкина "Образование на русском" https://pushkininstitute.ru/;
8. Справочно-информационный портал "Русский язык" http://gramota.ru/;
9. Служба тематических толковых словарей http://www.glossary.ru/;
10. Словари и энциклопедии http://dic.academic.ru/;
11. Образовательный портал "Учеба" http://www.ucheba.com/;
12. Законопроект "Об образовании в Российской Федерации". Вопросы и ответы http://xn--273--84d1f.xn--p1ai/voprosy_i_otvety

Собственные электронные образовательные и информационные ресурсы КубГУ

1. Электронный каталог Научной библиотеки КубГУ http://megapro.kubsu.ru/MegaPro/Web
2. Электронная библиотека трудов ученых КубГУ http://megapro.kubsu.ru/MegaPro/UserEntry?Action=ToDb&idb=6
3. Среда модульного динамического обучения http://moodle.kubsu.ru
4. База учебных планов, учебно-методических комплексов, публикаций и конференций http://infoneeds.kubsu.ru/
5. Библиотека информационных ресурсов кафедры информационных образовательных технологий http://mschool.kubsu.ru;
6. Электронный архив документов КубГУ http://docspace.kubsu.ru/
7. Электронные образовательные ресурсы кафедры информационных систем и технологий в образовании КубГУ и научно-методического журнала "ШКОЛЬНЫЕ ГОДЫ" http://icdau.kubsu.ru/

5.5 Публикации конференций A*

1. Höller, D., Behnke, G., Bercher, P., Biundo-Stephan, S., Fiorino, H., Pellier, D., & Alford, R. (2020). HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. AAAI Conference on Artificial Intelligence.

2. Feng, F., & Magliacane, S. (2023). Learning Dynamic Attribute-factored World Models for Efficient Multi-object Reinforcement Learning. Neural Information Processing Systems, abs/2307.09205.

3. Zan, D., Chen, B., Yang, D., Lin, Z., Kim, M., Guan, B., Wang, Y., Chen, W., & Lou, J. (2022). CERT: Continual Pre-Training on Sketches for Library-Oriented Code Generation. International Joint Conference on Artificial Intelligence, abs/2206.06888.

4. Cipriano, B.P., & Alves, P. (2024). LLMs Still Can't Avoid Instanceof: An Investigation Into GPT-3.5, GPT-4 and Bard's Capacity to Handle Object-Oriented Programming Assignments. 2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), 162-169.

5.4 Перечень информационно-коммуникационных технологий

1. Облачные платформы и сервисы
cloud.ru,YandexCloud,AWS/GCP/Azure– облачные вычисления

2. Системы управления версиями и коллаборации
Git/GitHub/GitLab – контроль версий кода и совместная разработка

4. Система управления обучением
Moodle – сдача работ

5.5 Перечень лицензионного и свободно распространяемого программного обеспечения

1. Свободное ПО (Open Source)
GitLab, GIT, Visual Studio Code.
2. Лицензионное ПО
Visual Studio, JetBrains Rider, SemanticKernel

6. Методические указания для обучающихся по освоению дисциплины (модуля)

По курсу предусмотрено проведение лекционных занятий, на которых дается систематизированный материал по объектно-ориентированному программированию. В ходе лекций рассматриваются ключевые концепции. После каждой лекции рекомендуется выполнение практических заданий для закрепления ключевых понятий и методов.

Лабораторные занятия курса посвящены практическому освоению методов объектно-ориентированного программирования. В ходе занятий разбираются готовые программные приложения и проводится анализ их построения. После занятия рекомендуется выполнить упражнения, приводимые в аудитории для самостоятельной работы.

При самостоятельной работе студентов необходимо изучить литературу, приведенную в перечнях выше, для осмысления вводимых понятий, анализа предложенных подходов и методов разработки программ. Разрабатывая решение новой задачи, студент должен уметь выбрать эффективные и надежные структуры данных для представления информации, подобрать соответствующие алгоритмы для их обработки, учесть специфику языка программирования, на котором будет выполнена реализация. Студент должен уметь выполнять тестирование и отладку алгоритмов решения задач с целью обнаружения и устранения в них ошибок.

Важнейшим этапом курса является самостоятельная работа по дисциплине. В процессе самостоятельной работы студент приобретает навыки объектно-ориентированного программирования.

Кейсы ПАО «Сбербанк»

8. Модель анализа инвестиционной привлекательности малого бизнеса
Описание:
Банк активно развивает кредитование и инвестиционные инструменты для малого и среднего предпринимательства (МСП). Требуется создать модель, которая на основе открытых и банковских данных (выручка, расходы, тип деятельности, отзывы, онлайн-активность) оценивает инвестиционную привлекательность МСП.
Цель:
Разработать систему рейтинговой оценки компаний малого бизнеса с возможностью визуализации факторов и динамики показателей.
Ожидаемый результат:
Модель, присваивающая компании инвестиционный рейтинг (например, A–E), объясняющая ключевые параметры и дающая рекомендации для инвестора.

9. Индивидуальная оценка кредитоспособности клиента на основе поведенческих данных

Описание:

Современный кредитный скоринг выходит за рамки финансовых данных. Необходимо исследовать, как поведенческие и цифровые следы (частота входа в мобильный банк, способы оплаты, география, время отклика) влияют на персональную оценку риска.

Цель:

Разработать ML-модель, оценивающую вероятность дефолта по нестандартным поведенческим признакам (возможно — с explainable AI).

Ожидаемый результат:

Прототип скоринговой модели, которая, помимо стандартных данных, учитывает цифровой профиль клиента и объясняет решения (SHAP, LIME и др.).

11. Анализ поведения пользователей в экосистеме цифрового рубля

Описание:

Сбербанк участвует в пилотных проектах по внедрению цифрового рубля. Интерес представляет исследование пользовательских паттернов: как изменяются модели потребления, скорости операций, уровень доверия, сравнение с классическим безналом.

Цель:

Построить модель анализа поведения клиентов, участвующих в транзакциях с цифровым рублем: частота, средний чек, контексты.

Ожидаемый результат:

Отчёт и ML-модель, классифицирующая типы пользователей и выявляющая ключевые различия в предпочтениях и барьерах цифровой валюты.

Кейсы от «АВАЛАБ»

2. Анализ обращений клиентов и CRM-переписки

Описание:

В службе клиентского сервиса застройщика ежедневно обрабатываются десятки обращений (e-mail, звонки, мессенджеры). Требуется реализовать систему семантического анализа и классификации NLU: выявлять суть обращений, уровень удовлетворенности, отслеживать повторяющиеся запросы.

Цель:

Автоматизировать первичный разбор и маршрутизацию запросов по тематике (сдача объекта, отделка, документы, жалоба и т.д.).

Ожидаемый результат:

Прототип, который выделяет суть обращений и формирует дашборд по текущим «болям» клиентов.

4. Генерация рекламного контента для жилых комплексов

Описание:

«АВА ГРУПП» регулярно запускает маркетинговые кампании для жилых комплексов. Необходимо исследовать использование диффузионных моделей для генерации изображений (визуализации интерьеров, окрестностей, видов из окон) и LLM — для описаний квартир, преимуществ района, инфраструктуры.

Цель:

Создать инструменты для быстрой генерации продающих материалов без привлечения дизайнеров и копирайтеров на первых этапах.

Ожидаемый результат:

Набор сгенерированных карточек объектов с текстом, изображением и логикой «живого» рекламного сообщения.

6. Генерация документации и шаблонов договоров

Описание:

Юридический департамент регулярно работает с договорами долевого участия, актами

приёма-передачи и другими документами. Использование LLM может значительно сократить время на подготовку черновиков — достаточно ввести параметры сделки.
Цель:
Создать систему, которая генерирует адаптированные тексты документов по вводным данным (тип объекта, этаж, площадь, ФИО, сроки и пр.).
Ожидаемый результат:
Генератор документов в формате Word или PDF с автоматической подстановкой параметров и соблюдением юридического стиля.

Кейс от ООО «СвязьРесурс-Кубань»
Описание:
Компания ООО "СвязьРесурс-Кубань" оказывает услуги связи. Работа с клиентами автоматизирована на базе CRM Битрикс 24. Для компании актуальны вопросы разработки первоначальных версий документов с помощью LLM и в перспективе автоматизации генерации большого количества документов по шаблонам с помощью LLM и RAG системы с интеграцией с Битрикс 24. Задачи включают в себя:
1. Разработка библиотеки промптов для генерации регламентов описания бизнес-процессов Битрикс 24.
2. Разработка библиотеки промптов для генерации техзаданий на основе параметров оказания услуг.
3. Разработка библиотеки промптов для генерации коммерческих предложений на основе параметров оказания услуг.
4. Разработка библиотеки промптов для генерации скриптов работы технической поддержки.
5. Разработка библиотеки промптов для генерации скриптов работы отдела продаж.
6. Апробация и сравнение различных языковых моделей для решения задач.
Цель:
Автоматизировать работу сотрудников по составлению типовых документов.
Ожидаемый результат:
Библиотека промптов и рекомендации по использованию LLM для решения поставленных задач.

Для студентов с ограниченными возможностями здоровья предусмотрены дополнительные индивидуальные консультации, на которых преподаватель подробно разъясняет сложные аспекты дисциплины, помогает адаптировать практические задания и обеспечивает специальные условия для освоения методов работы с системами искусственного интеллекта. Индивидуальный подход позволяет таким студентам полноценно участвовать в учебном процессе и достигать требуемых результатов обучения.

7. Материально-техническое обеспечение по дисциплине (модулю)

Виртуальные машины, кластер Managed Kubernetes и ресурсы GPU в облаке предоставляется индустриальным партнером ПАО «Сбербанк»:

№	Продукт	Параметры продукта	Кол-во	Кол-во конфигураций	Ед. изм.

1	Виртуальная машина	Виртуальная машина 10% vCPU 2 vCPU 4 RAM	1	60	Шт
		ОС Ubuntu 22.04	1		Шт
		Системный диск SSD	1		Шт
			10		Гб
		Аренда публичного IP	1		Шт
2	Виртуальная машина с GPU	Виртуальная машина с GPU NVIDIA® Tesla® V100 2 GPU 8 vCPU 128 ГБ RAM	1	1	Шт
		ОС Ubuntu_24.04	1		Шт
		Системный диск SSD	1		Шт
			2000		Гб
		Диск SSD	1		Шт
			4096		Гб
		Диск SSD	1		Шт
			4096		Гб
		Аренда публичного IP	1		Шт
3	K8S	Master node 8 vCPU 16 RAM	1	1	Шт
		Worker node 10% доля 4 vCPU 32 RAM	5		Шт
		Worker node SSD-NVME	64		Гб
		Аренда публичного IP	1		Шт
4	ML Inference Instance Type GPU	Время работы в месяц	40	1	Ч
		Инстанс 8 x NVIDIA® H100 NVLink PCIe 160 vCPU 1520 GB RAM	1		Шт
		Количество запросов к ML-моделям	1		Млн. Шт
		Кэш ML-моделей	160		Гб
5	LLM	Токены GigaChat 2 Max	50		Млн. Шт
		Токены Embeddings	400		Млн. Шт

Дополнительные облачные ресурсы предоставляются технологическим партнером Yandex Cloud.

№	Вид работ	Наименование учебной аудитории, ее оснащенность оборудованием и техническими средствами обучения
1.	Лекционные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения
2.	Лабораторные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, проектором, программным обеспечением

3.	Групповые (индивидуальные) консультации	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
4.	Текущий контроль, промежуточная аттестация	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
5.	Самостоятельная работа	Кабинет для самостоятельной работы, оснащенный компьютерной техникой с возможностью подключения к сети «Интернет», программой экранного увеличения и обеспеченный доступом в электронную информационно-образовательную среду университета.

Примечание: Конкретизация аудиторий и их оснащение определяется ОПОП.