

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет
им. Н.И. Лобачевского»**

Институт информационных технологий, математики и механики

УТВЕРЖДЕНО
решением Ученого совета ННГУ
протокол № 10 от 27.08.2025 г.

Рабочая программа дисциплины

Разработка сетевых приложений на Java

Уровень высшего образования
Бакалавриат

Направление подготовки / специальность
01.03.02 - Прикладная математика и информатика

Направленность образовательной программы
Математическое моделирование и искусственный интеллект

Форма обучения
очная

г. Нижний Новгород

2025 год начала подготовки

1. Место дисциплины в структуре ОПОП

Дисциплина Б1.В.16 Разработка сетевых приложений на Java относится к части, формируемой участниками образовательных отношений образовательной программы.

2. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы (компетенциями и индикаторами достижения компетенций)

Формируемые компетенции (код, содержание компетенции)	Планируемые результаты обучения по дисциплине (модулю), в соответствии с индикатором достижения компетенции		Наименование оценочного средства	
	Индикатор достижения компетенции (код, содержание индикатора)	Результаты обучения по дисциплине	Для текущего контроля успеваемости	Для промежуточной аттестации
<p><i>ПК-КРМ_3:</i> Способен выполнять интеграцию программных модулей и компонентов и проверять работоспособность выпусков программного продукта</p>	<p><i>ПК-КРМ_3.1:</i> Разрабатывает процедуры интеграции программных модулей</p> <p><i>ПК-КРМ_3.2:</i> Выполняет интеграцию программных модулей и компонентов и проверку работоспособности выпусков программного продукта</p>	<p><i>ПК-КРМ_3.1:</i> <i>ПК-КРМ_3.1. 3-1.</i> Знает методы и средства сборки модулей и компонентов компьютерного программного обеспечения.</p> <p><i>ПК-КРМ_3.1. 3-2.</i> Знает интерфейсы взаимодействия с внешней средой.</p> <p><i>ПК-КРМ_3.1. 3-3.</i> Знает интерфейсы взаимодействия внутренних модулей системы.</p> <p><i>ПК-КРМ_3.1. У-1.</i> Умеет писать программный код процедур интеграции программных модулей.</p> <p><i>ПК-КРМ_3.1. У-2.</i> Умеет использовать выбранную среду программирования для разработки процедур интеграции программных модулей.</p> <p><i>ПК-КРМ_3.1. У-3.</i> Умеет применять методы и средства сборки модулей и компонентов компьютерного программного обеспечения, разработки процедур для развертывания компьютерного программного обеспечения, миграции и преобразования данных, создания программных интерфейсов.</p> <p><i>ПК-КРМ_3.2:</i></p>	Задачи	Зачёт: Контрольные вопросы

		<p>ПК-КРМ_3.2. 3-1. Знает методы и средства сборки и интеграции программных модулей и компонентов.</p> <p>ПК-КРМ_3.2. 3-2. Знает методы и средства проверки работоспособности выпусков программных продуктов.</p> <p>ПК-КРМ_3.2. У-1. Умеет выполнять процедуры сборки программных модулей и компонентов в программный продукт.</p> <p>ПК-КРМ_3.2. У-2. Умеет производить настройки параметров программного продукта и осуществлять запуск процедур сборки.</p> <p>ПК-КРМ_3.2. У-2. Умеет проводить проверку работоспособности программного продукта.</p>		
<p>ПК-Опер_1: Осуществляет управление архитектурой изолированной (неинтегрированной) программной системы</p>	<p>ПК-Опер_1.1: Выявляет и согласовывает требования к программной системе с точки зрения архитектуры</p> <p>ПК-Опер_1.2: Осуществляет выбор и моделирование архитектурного решения для реализации программной системы</p> <p>ПК-Опер_1.3: Разрабатывает разделы по архитектуре проектных и эксплуатационных документов программной системы</p> <p>ПК-Опер_1.4: Контролирует реализацию и испытания программной системы с точки зрения архитектуры</p> <p>ПК-Опер_1.5: Осуществляет сопровождение эксплуатации программной системы с точки зрения архитектуры</p>	<p>ПК-Опер_1.1:</p> <p>ПК-Опер_1.1. У-1. Способен выявлять несоответствия требований заказчика к программной системе с точки зрения архитектуры</p> <p>ПК-Опер_1.1. У-2. Способен описывать требования к программной системе с точки зрения архитектуры</p> <p>ПК-Опер_1.1. У-3. Умеет проверять требования на соответствие архитектуре программной системы</p> <p>ПК-Опер_1.1. У-4. Умеет выявлять требования к архитектуре программной системы путем проведения интервью с заинтересованными сторонами</p> <p>ПК-Опер_1.1. У-5. Умеет формулировать архитектурные требования к программной системе</p> <p>ПК-Опер_1.1. 3-1. Знает методы управления требованиями.</p> <p>ПК-Опер_1.1. 3-2. Знает методы моделирования архитектуры программной</p>	Задания	<p>Зачёт: Контрольные вопросы</p>

		<p>системы</p> <p>ПК-Опер_1.1. 3-3. Знает методы проектирования архитектуры программной системы</p> <p>ПК-Опер_1.2:</p> <p>ПК-Опер_1.2. У-1. Способен выбрать оптимальное архитектурное решение с учетом особенностей программной системы и принципов её организации</p> <p>ПК-Опер_1.2. У-2. Способен определить архитектуру системы, ее, бизнес-процессов, структуру данных и отдельных компонентов программной системы и методы их интеграции.</p> <p>ПК-Опер_1.2. У-3. Способен определить перечень элементов архитектуры, которые должны быть защищены от угроз безопасности информации, связанных с нарушением конфиденциальности, целостности и доступности.</p> <p>ПК-Опер_1.2. У-4. Способен моделировать архитектурное решение для изолированной программной системы</p> <p>ПК-Опер_1.2. У-5. Умеет проектировать бизнес-архитектуру программных систем с применением лучших практик, шаблонов и стилей архитектурного проектирования</p> <p>ПК-Опер_1.2. 3-2. Знает протоколы взаимодействия программных систем</p> <p>ПК-Опер_1.2. 3-3. Знает нормативные правовые акты, организационно-распорядительные документы и методические рекомендации, определяющие требования к безопасности программного обеспечения</p> <p>ПК-Опер_1.3:</p>		
--	--	---	--	--

		<p><i>ПК-Опер_1.3. У-1. Способен описывать технические и организационные меры, обеспечивающие сохранение и восстановление программного обеспечения</i></p> <p><i>ПК-Опер_1.3. У-2. Умеет проектировать и моделировать архитектурные элементы программных систем и их взаимосвязи.</i></p> <p><i>ПК-Опер_1.3. У-3. Умеет формировать технические и организационные меры для защиты программной системы от несанкционированного доступа к элементам конфигурации</i></p> <p><i>ПК-Опер_1.3. З-1. Знает методы моделирования и технического описания архитектуры программных систем</i></p> <p><i>ПК-Опер_1.3. З-2. Знает нормативные правовые акты, организационно-распорядительные документы и методические рекомендации, определяющие требования к безопасности программного обеспечения</i></p> <p><i>ПК-Опер_1.4:</i></p> <p><i>ПК-Опер_1.4. У-1. Способен проверять соответствие реализации программной системы выбранному архитектурному решению</i></p> <p><i>ПК-Опер_1.4. У-2. Способен проверять результаты испытаний программной системы на соответствие архитектуре и архитектурным решениям</i></p> <p><i>ПК-Опер_1.4. У-3. Умеет проверять характеристики реализованной программной системы на соответствие архитектурным требованиям</i></p> <p><i>ПК-Опер_1.4. З-1. Знает способы определения характеристик работающей программной системы</i></p>		
--	--	---	--	--

		<p>ПК-Опер_1.4. 3-2. Знает методы параметризации архитектуры программных систем</p> <p>ПК-Опер_1.4. 3-3. Знает</p> <p>ПК-Опер_1.5:</p> <p>ПК-Опер_1.5. У-1. Способен проверять запросы на изменения программной системы на реализуемость с точки зрения архитектуры программной системы</p> <p>ПК-Опер_1.5. У-2. Способен согласовывать запросы на изменения программной системы с точки зрения архитектуры</p> <p>ПК-Опер_1.5. У-3. Умеет взаимодействовать с авторами запросов на изменения программной системы для уточнения содержания запросов</p> <p>ПК-Опер_1.5. 3-1. Знает основы процесса управления изменениями программных систем.</p> <p>ПК-Опер_1.5. 3-2. Знает методы обеспечения устойчивости функционирования программной системы.</p> <p>ПК-Опер_1.5. 3-3. Знает методы обеспечения надежности архитектуры программной системы.</p>		
<p>ПК-ТОП_11: Способен разрабатывать, оптимизировать и отлаживать системное программное обеспечение</p>	<p>ПК-ТОП_11.1: Разрабатывает низкоуровневый код для встраиваемых систем и драйверов</p> <p>ПК-ТОП_11.2: Оптимизирует код под ограниченные ресурсы</p> <p>ПК-ТОП_11.3: Работает с ОС, загрузчиками и аппаратурой</p> <p>ПК-ТОП_11.4: Отлаживает системное ПО без ОС</p>	<p>ПК-ТОП_11.1: ПК-ТОП_11.1. 3-1. Знает архитектуру современных процессоров, особенности работы с оборудованием процессора</p> <p>ПК-ТОП_11.1. 3-2. Знает принципы взаимодействия ПО с аппаратурой</p> <p>ПК-ТОП_11.1. У-1. Умеет разрабатывать низкоуровневый код для встроенного программного обеспечения и драйверов</p> <p>ПК-ТОП_11.1. У-2. Умеет</p>	Задания	<p>Зачёт: Контрольные вопросы</p>

		<p>разрабатывать драйверы для аппаратных устройств.</p> <p><i>ПК-ТОП_11.2:</i> <i>ПК-ТОП_11.2. 3-1. Знает методы оптимизации (кэш, память, тактовая частота процессора).</i> <i>ПК-ТОП_11.2. 3-2. Знает инструменты профилирования.</i> <i>ПК-ТОП_11.2. У-1. Умеет анализировать бенчмарки, выявлять узкие места.</i> <i>ПК-ТОП_11.2. У-2. Умеет разрабатывать код, оптимизированный для выбранной аппаратной архитектуры.</i></p> <p><i>ПК-ТОП_11.3:</i> <i>ПК-ТОП_11.3. 3-1. Знает архитектуру ядра Linux.</i> <i>ПК-ТОП_11.3. 3-2. Знает принципы работы встраиваемых операционных систем.</i> <i>ПК-ТОП_11.3. У-1. Умеет портировать код между различными вариантами загрузчиков и ОС.</i> <i>ПК-ТОП_11.3. У-2. Умеет выполнять разработку на стыке программного обеспечения и оборудования.</i></p> <p><i>ПК-ТОП_11.4:</i> <i>ПК-ТОП_11.4. 3-1. Знает методы анализа bare-metal сбоев</i> <i>ПК-ТОП_11.4. 3-2. Знает форматы бинарных файлов, работу с памятью.</i> <i>ПК-ТОП_11.4. У-1. Умеет отлаживать код через специализированные отладчики системного программного обеспечения.</i> <i>ПК-ТОП_11.4. У-2. Умеет анализировать дампы памяти, дизассемблированный код.</i></p>		
--	--	--	--	--

<p>ППК-Р6: Способен участвовать в промышленной разработке программного обеспечения (top)</p>	<p>ППК-Р6.1: Работает в соответствии с промышленными методологиями разработки</p> <p>ППК-Р6.2: Использует инструменты промышленной разработки</p> <p>ППК-Р6.3: Разрабатывает масштабируемый и поддерживаемый код</p>	<p>ППК-Р6.1:</p> <p>ППК-Р6.1. 3-1. Знает принципы Agile и их применение в промышленных проектах.</p> <p>ППК-Р6.1. 3-2. Знает процессы code review, принципы коллективного владения кодом (collective code ownership).</p> <p>ППК-Р6.1. У-1. Умеет оценивать объем задачи и срок ее выполнения, участвовать в планировании спринтов.</p> <p>ППК-Р6.1. У-2. Умеет работать в команде с использованием инструментов управления проектами.</p> <p>ППК-Р6.2:</p> <p>ППК-Р6.2. 3-1. Знает принципы Continuous Integration and Continuous Delivery (CI/CD).</p> <p>ППК-Р6.2. 3-2. Знает системы мониторинга и логирования в продуктивной среде.</p> <p>ППК-Р6.2. У-1. Умеет настраивать потоки работ CI/CD.</p> <p>ППК-Р6.2. У-2. Умеет работать с контейнеризацией и оркестрацией.</p> <p>ППК-Р6.2. У-2. Умеет настраивать мониторинг в продуктивной среде.</p> <p>ППК-Р6.3:</p> <p>ППК-Р6.3. 3-1. Знает принципы чистого кода, SOLID, DRY, KISS и др.</p> <p>ППК-Р6.3. 3-2. Знает принципы предметно-ориентированного проектирования (ПОП) программного обеспечения.</p> <p>ППК-Р6.3. 3-3. Знает паттерны проектирования и антипаттерны.</p> <p>ППК-Р6.3. У-1. Умеет разрабатывать модульный и</p>	<p>Задания</p>	<p>Зачёт: Контрольные вопросы</p>
--	--	---	----------------	---------------------------------------

		<p>тестируемый программный код.</p> <p>ППК-Р6.3. У-2. Умеет выполнять модульное, интеграционное и нагрузочное тестирование.</p> <p>ППК-Р6.3. У-3. Умеет проводить рефакторинг для повышения качества кода.</p>		
--	--	--	--	--

3. Структура и содержание дисциплины

3.1 Трудоемкость дисциплины

	очная
Общая трудоемкость, з.е.	2
Часов по учебному плану	72
в том числе	
аудиторные занятия (контактная работа):	
- занятия лекционного типа	12
- занятия семинарского типа (практические занятия / лабораторные работы)	24
- КСР	1
самостоятельная работа	35
Промежуточная аттестация	0
	Зачёт

3.2. Содержание дисциплины

(структурированное по темам (разделам) с указанием отведенного на них количества академических часов и виды учебных занятий)

Наименование разделов и тем дисциплины	Всего (часы)	в том числе			Самостоятельная работа обучающегося, часы
		Контактная работа (работа во взаимодействии с преподавателем), часы из них			
		Занятия лекционного типа	Занятия семинарского типа (практические занятия/лабораторные работы), часы	Всего	
Ф	Ф	Ф	Ф	Ф	
Введение в Java (компиляция и запуск приложений)	6	1	2	3	3
Повторение основ ООП	6	1	2	3	3
Расширенные возможности с точки зрения синтаксиса Java	6	1	2	3	3
Коллекции в Java (Stream API)	6	1	2	3	3
Демонстрация выполненного учебного примера	6	1	2	3	3
Разработка визуальных приложений	7	1	2	3	4
Java и элементы многопоточного программирования	8	1	3	4	4

Разработка сетевых приложений на основе Socket	8	1	3	4	4
Организация взаимодействия с базами данных в Java	9	2	3	5	4
Разработка мобильных приложений на основе Android	9	2	3	5	4
Аттестация	0				
КСР	1			1	
Итого	72	12	24	37	35

Содержание разделов и тем дисциплины

Цели и задачи изучения дисциплины.

Целями освоения дисциплины «Разработка сетевых приложений на Java» являются освоение современных технологий разработки программного обеспечения с применением сетевых технологий взаимодействия программ на примере языка программирования Java, а также формирование у обучающихся практических навыков создания клиент-серверных, многопоточных и высокопроизводительных приложений с поддержкой баз данных.

Задачами изучения дисциплины «Разработка сетевых приложений на Java» являются:

1. Освоение основ многопоточного программирования в Java, включая создание и управление потоками, механизмы синхронизации доступа к общим ресурсам, а также практическое применение высокоуровневых конструкций из пакета `java.util.concurrent` (пулы потоков, блокирующие очереди, атомарные переменные) для повышения производительности, отказоустойчивости и масштабируемости приложений.
2. Изучение принципов разработки сетевых приложений на основе сокетов, обеспечивающих надежный обмен данными между клиентом и сервером по протоколу TCP/IP; освоение создания многопользовательских серверов, обработки входящих соединений в отдельных потоках, протоколирования сетевого обмена, а также корректного установления и завершения сетевых соединений.
3. Овладение технологиями взаимодействия Java с реляционными базами данных, включая низкоуровневую работу через JDBC (установка соединения, выполнение запросов, обработка результатов, управление транзакциями и обработка исключений) и использование высокоуровневого фреймворка Hibernate (ORM: отображение сущностей на таблицы, HQL-запросы, кэширование, управление жизненным циклом объектов) для упрощения и ускорения разработки слоя доступа к данным.
4. Закрепление полученных знаний путем разработки и защиты полноценного учебного приложения, интегрирующего сетевые возможности, многопоточность и работу с базами данных, что позволяет продемонстрировать умение применять современные технологии комплексно для решения реальных практических задач.

Для достижения поставленных целей и задач изучаются следующие лекционные материалы:

1. Введение в Java (компиляция и запуск приложений)
2. Повторение основ ООП
3. Расширенные возможности с точки зрения синтаксиса Java
4. Коллекции в Java (Stream API)
5. Демонстрация выполненного учебного примера
6. Разработка визуальных приложений
7. Java и элементы многопоточного программирования
8. Разработка сетевых приложений на основе Socket

9. Организация взаимодействия с базами данных в Java
10. Разработка мобильных приложений на основе Android

4. Учебно-методическое обеспечение самостоятельной работы обучающихся

Самостоятельная работа обучающихся включает в себя подготовку к контрольным вопросам и заданиям для текущего контроля и промежуточной аттестации по итогам освоения дисциплины приведенным в п. 5.

Для обеспечения самостоятельной работы обучающихся используются:
Открытые онлайн-курсы MOOC:

Базовый курс по Java, e-learning.unn.ru/course/view.php?id=9771.

Иные учебно-методические материалы:

1. Специализация ObjectOrientedProgramminginJava.–Coursera.–
<https://www.coursera.org/specializations/object-oriented-programming>.
2. Серия обучающих материалов по Java.– <https://dev.java/learn/>.
3. JavaPlatform, StandardEdition Documentation.– <https://docs.oracle.com/en/java/javase/>.
4. Руководство пользователя по технологии Spring.–<https://spring.io/guides>.
5. Руководство пользователя по технологиям Spring и Hibernate.–<https://www.baeldung.com/>.
6. Распределенное программирование на языке Java.–Coursera.–
<https://www.coursera.org/learn/distributed-programming-in-java>.
7. Введение в Maven.–<https://maven.apache.org/guides/getting-started/index.html>.
8. Введение в Gradle.–<https://gradle.org/guides/>.
9. Введение в Ant.–<https://ant.apache.org/manual/tutorial-HelloWorldWithAnt.html>.
10. Домашняя страница технологии JavaFX.–<https://openjfx.io/>.

5. Фонд оценочных средств для текущего контроля успеваемости и промежуточной аттестации по дисциплине (модулю)

5.1 Типовые задания, необходимые для оценки результатов обучения при проведении текущего контроля успеваемости с указанием критериев их оценивания:

5.1.1 Типовые задания (оценочное средство - Задачи) для оценки сформированности компетенции ПК-КРМ_3:

Пример технического задания к лабораторным работам

Клиент-серверная игра «меткий стрелок»

Общий вид интерфейса:

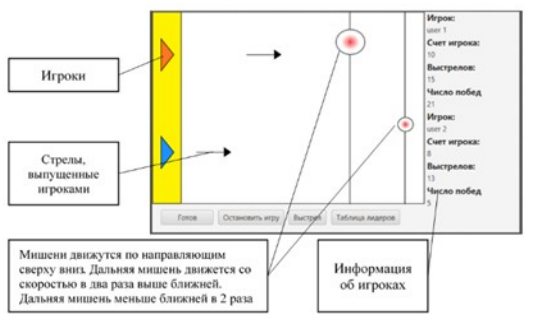


Рис. 1. Интерфейс клиент-серверного приложения.

Требования к реализации

1. Язык программирования Java.
2. Визуальный интерфейс реализован с использованием Swing или JavaFX.
3. Анимация в игре реализуется на основе класса Thread.
4. Реализация взаимодействия клиентов на основе классов Socket, ServerSocket
5. Взаимодействие с базой данных реализовано с использованием технологии Hibernate

Сторона сервера:

1. Сервер ожидает подключения игроков
 - a. Максимальное число игроков 4.
2. Сервер содержит информацию об игроках
 - a. Имя пользователя
 - b. Счет
 - c. Число выстрелов
 - d. Число побед
3. Сервер контролирует процесс игры
 - a. Перемещение мишеней
 - b. Положение стрел
 - c. Расчёт числа попаданий
4. Имена пользователей и число побед должны храниться в базе данных. Сервер должен обеспечивать доступ к соответствующей информации.

Сторона клиента:

1. Клиент отображает состояние игры (рис. 1)
2. Клиент может подключиться к серверу задав имя пользователя.
 - a. Имена пользователей не должны повторяться
3. После подключения у клиента есть возможности
 - a. Подтвердить готовность начать игру
 - b. Сказать о необходимости приостановить игру
 - c. Сделать выстрел
 - d. Получить таблицу лидеров. В таблице отображается две колонки – имя пользователя и число побед

Общие положения к работе клиент-серверного приложения:

1. Запускается сервер.
2. Запускаются клиентские приложения.
3. Клиенты подключаются к серверу.
4. Клиенты сообщают о готовности начать игру.

5. Если все игроки готовы, сервер запускает игру начиная перемещать мишени и сбросив набранные очки всех игроков.
6. Игроки в процессе игры могут:
 - а. Делать выстрел.
 - б. Сообщать о паузе. Игра приостанавливается. Чтобы снять паузу игрок должен снова сообщить о готовности.
7. Игра завершается при наборе одним из игроков 6 очков. Тот, кто первый набрал очки считается победителем. Имя победителя сообщается всем игрокам. Игра завершается.
8. После завершения игры должна быть возможность перейти к пункту 4.
9. В любой момент после подключения к серверу клиент должен иметь возможность получить таблицу лидеров.

Вариант 1. Эмулятор сетей

Необходимо реализовать эмулятор сетей с устройствами первых трех уровней. Т.е. сетевые карты, коммутаторы и концентраторы, а так же роутеры. В эмуляторе у пользователей должна быть предусмотрена возможность совместного редактирования и настройки виртуальной сети. Созданные топологии сетей должны храниться в базе данных. С помощью полученной топологии пользователи должны иметь возможность проверить корректность настройки сетевых устройств и сбора статистики. Для проверки необходимо реализовать на сетевых картах консоль с командами ipconfig, trace и ping. В качестве статистики необходимо выдавать количество пройденных сетевых устройств и длину пути.

Вариант 2. Удаленная виртуальная машина (разработка кода)

Необходимо реализовать клиент серверное приложение. В клиенте для пользователя должна быть предусмотрена возможность вводить программу на псевдоязыке программирования. Разработанная на клиенте программа должна приниматься и исполняться на сервере. На клиенте реализовать возможность совместного редактирования кода. При изменении кода одним из пользователей изменения должны видеть все пользователи. Программы, разработанные пользователями, должны храниться в базе данных. Реализовать возможность сохранения изменений в коде в виде патчей (<https://code.google.com/archive/p/java-diff-utils/>). Для каждого патча реализовать возможность обсуждения и отката.

Вариант 3. Удаленная виртуальная машина (исполнение кода)

Необходимо реализовать клиент серверное приложение. В клиенте для пользователя должна быть предусмотрена возможность вводить программу на псевдоязыке программирования. Разработанная на клиенте программа должна приниматься и исполняться на сервере. В один и тот же момент времени может исполняться программ. На клиенте реализовать возможность отображения очереди задач и истории запусков. Очередь задач и история запусков должна храниться в базе данных. Пользователь должен иметь возможность по истории запусков восстановить код и результаты экспериментов. С каждым запуском должна быть реализована возможность обсуждения результатов.

Вариант 4. Удаленная виртуальная машина (отладка)

Необходимо реализовать клиент серверное приложение. В клиенте для пользователя должна быть предусмотрена возможность вводить программу на псевдоязыке программирования. Разработанная на клиенте программа должна приниматься и исполняться на сервере. На клиенте реализовать возможность совместной отладки кода (исполнение по шагам, исполнение до контрольной точки, вывод значений переменных). Каждый клиент должен иметь возможность оставлять комментарии к коду и возможность изменения текущего состояния программы. Программы и комментарии к коду должны храниться в базе данных.

Вариант 5. Удаленное управление и запуск задач

Необходимо разработать клиент-серверное приложение удаленного управления и запуска задач. Клиент должен позволять загружать исполняемые программы на сервер, обозревать и скачивать результирующие файлы. Пользователь должен иметь возможность задавать серию экспериментов. В серии экспериментов пользователь должен иметь возможность устанавливать зависимости между задачами. Описание задач и серий экспериментов должны храниться в базе данных. При задании задачи или серии экспериментов информация о запусках передается на сервер. После отправки задания пользователь может отключиться от сервера. На клиенте реализовать возможность отображения очереди задач. При подключенном клиенте к серверу пользователь должен автоматически получать информацию о состоянии очереди задач. В один и тот же момент времени может исполняться программ.

Вариант 6. Многопользовательская игра «пакмен»

Разработать многопользовательскую игру «Пакмен». Серверная часть программы должна включать отслеживание положения игроков и разрешать вопросы их взаимодействия. Клиентская часть программы должна обеспечивать возможность подключения к игре, отображение поля игры и возможность управлять своим «пакменом». Предполагается, что на сервере реализована поддержка создания сессий игр. В каждую сессию могут входить до четырех игроков и произвольное количество зрителей. Кроме игроков в игре должны присутствовать «приведения» охотящиеся за игроками. «Приведениями» управляют клиентские программы с искусственным интеллектом. Информация о проведенных играх должна сохраняться в базе данных.

Вариант 7. Многопользовательская гонка с препятствиями

Разработать многопользовательскую игру гонку. В игре трасса отображается видом сверху и прокручивается сверху вниз. В процессе гонки на трассе появляются препятствия. При столкновении с препятствиями игрок выбывает. Игрок может перемещаться по трассе влево и вправо. Цель каждого игрока продержаться на трассе дольше соперников. В каждой гонке может участвовать до четырех участников и произвольное количество зрителей. Информацию об игроках и их победах хранить в базе данных.

Вариант 8. Распределенный визуальный редактор

Разработать клиент-серверную программу, которая бы позволила пользователям одновременно создавать и редактировать изображения. В качестве базовых объектов используются точки, линии, прямоугольники, дуги и окружности. Реализовать возможность конструирования более сложных объектов из базовых. Обеспечить возможность хранения и повторного использования сложных объектов в базе данных. Для объектов реализовать возможность перемещения и вращения. Реализовать возможность взаимоисключения одновременного редактирования пользователями одних и тех же объектов. Реализовать возможность простейшей анимации объектов.

Вариант 9. Программа общения пользователей с расширенными возможностями планирования

Разработать программу обмена мгновенными сообщениями и планирования. Пользователь должен уметь создавать задачи и назначать их себе или участникам беседы. Задачи могут быть описаны текстом с картинками. В клиентской программе должно быть предусмотрено два типа обмена сообщениями: частный обмен сообщениями и конференция между несколькими участниками. Обмен сообщениями должен служить обсуждению созданных задач. Список задач должен быть доступен на любом клиенте пользователя. Реализовать фильтр задач по участникам встреч. История общения и список созданных задач должны храниться в базе данных.

Вариант 10. Распределенный администратор сайта.

Разработать программу, в которой администраторы совместно могли бы редактировать html код страниц сайтов. Программа «администратор» должна отображать информацию о том, кто из пользователей и

какие страницы редактирует. При одновременном редактировании файла реализовать возможность блокирования доступа к части документа. Клиент должен подсвечивать заблокированный текст и отображать информацию о редакторе. Редактировать можно только заблокированный участок текста. После снятия ограничений изменения должны сохраняться в редактируемом файле и обновиться у клиентов. Историю изменений страниц хранить в базе данных.

Вариант 11. Игра в балансир.

Разработать многопользовательскую игру в балансир. Число игроков 2-4. Каждый из игроков получает по 5 грузов, которые можно по одному размещать на шаткой платформе, подвешенной за центр. Грузы всех игроков видны всем участникам. Балансир свободно качается, реагируя на малейшее изменение веса. Игроки ходят одновременно, выбирая, куда и когда поставить свой груз. Если в момент установки груза платформа кренился больше чем на 20 градусов, наступает обвал. Проигрывает тот, чей груз стал последним при обвале. Победа наступает при установке всех грузов или проигрыше других участников. Положение платформы рассчитывает сервер по упрощенной модели.

Вариант 12. Игра в «вышибалы»

Разработать многопользовательскую игру, где участники сражаются на ограниченной арене, пытаясь выбить соперников с поля с помощью мячей. Число клиентов 2-4. Каждый игрок получает 5 мячей, которые можно бросать в соперников. Мячи и сами игроки видны всем. Игрок, попавший по сопернику, остаётся в игре, а тот, в кого попали, выбывает. На арене могут быть простые препятствия, представленные геометрическими фигурами. Положением игрока может меняться нажатием кнопок. Сервер отслеживает состояние всех участников: их позиции, количество оставшихся мячей, статус (в игре / выбыл), а также обрабатывает броски, попадания и столкновения.

Вариант 13. Игра в волейбол.

Разработать 2D многопользовательский волейбол. Число клиентов 2-4. Цель - перебрасывать мяч через «сетку» так, чтобы соперник не смог его отбить. Сетка расположена по середине игрового поля. Мяч летит по прямой. Каждая команда может касаться мяча не более трёх раз перед тем, как отправить его на сторону противника. Игроки не имеют права переходить через центральную линию, но на своем поле могут перемешаться, как угодно. Побеждает та команда, которая первой наберёт 2 очка. Сервер отслеживает положение игроков и мяча, проверяет правила, определяет очки.

Вариант 14. 2D многопользовательская игра: «Световые ловушки»

Разработать многопользовательскую игру, где игроки соревнуются в темноте. У каждого участника есть фонарь, освещающим небольшую зону перед собой. Видно только то, что попадает в поле зрения фонаря. Число клиентов 2-4. Цель - поймать других игроков в свой свет, но при этом не попасть в чужой луч. Как только игрок оказывается в свете соперника сразу проигрывает. Побеждает тот, кто первым поймает всех остальных игроков. На поле могут быть простые препятствия, представленные геометрическими фигурами. Сервер хранит полную карту, но клиенты получают только ту информацию, которая «в свете». Возможно усложнение игры Каждый игрок может кратковременно 2 раза включать режим «вспышки» - на секунду расширяет радиус света, но после этого не может ходить заданное время.

Критерии оценивания (оценочное средство - Задачи)

Оценка	Критерии оценивания
зачтено	Выполнена основная часть задания, возможно с незначительными недочетами.
не зачтено	Выполнено менее половины задания, есть существенные недочеты.

5.1.2 Типовые задания (оценочное средство - Задания) для оценки сформированности компетенции ПК-Опер_1:

Задачи для освоения компетенции ПК-Опер_1 используются из раздела 5.1.1. Компетенция ПК-Опер_1 формируется на основании следующих заданий:

1. Требования.

На основе выбранного варианта из раздела 5.1.1 составьте и согласуйте с преподавателем список архитектурно значимых требований к приложению. Опираясь на пример технического задания сформируйте 10–15 требований с разделением на категории: функциональные, нефункциональные, технологические ограничения. Результат предоставьте в виде таблицы. На защите устно обоснуйте 2–3 ключевых требования.

2. Архитектурное решение.

Выберите и обоснуйте архитектуру приложения: топологию (клиент-сервер), многопоточную модель сервера, текстовый протокол поверх сокетов, способ работы с базой данных. Нарисуйте две схемы: диаграмму компонентов и диаграмму развёртывания. Подготовьте краткое обоснование разработанной архитектуры.

3. Документация.

Разработайте два коротких раздела документации: «Архитектура программной системы» и «Руководство администратора». Раздел «Архитектура программной системы» должен содержать описание структуры классов, потоков данных и взаимодействия компонентов. Раздел «Руководство администратора» должен содержать описание методов запуска сервера и настройки БД, просмотра подключённых клиентов, а также корректного завершения работы.

4. Контроль и испытания.

Разработайте план испытаний, включающий проверку соответствия архитектуре, нагрузочное тестирование с 4 клиентами, а также проверку отказоустойчивости (например, обрыв соединения и пауза). Проведите испытания и продемонстрируйте преподавателю два теста: один нагрузочный и один отказоустойчивый.

5. Сопровождение.

После сдачи основной работы исправьте замечания, выявленные преподавателем, и добавьте дополнительный атрибут качества (например, логирование, автосохранение, мониторинг и т.д.). Ведите журнал сопровождения разработки, в котором фиксируйте дату, тип события, описание и решение. На защите покажите журнал и продемонстрируйте одно исправленное замечание.

5.1.3 Типовые задания (оценочное средство - Задания) для оценки сформированности компетенции ПК-ТОП_11:

Задачи для освоения компетенции ПК-ТОП_11 используются из раздела 5.1.1. Компетенция ПК-ТОП_11 формируется на основании следующих заданий:

1. Анализ сетевого трафика.

На основе выбранного варианта из раздела 5.1.1 проведите анализ сетевого взаимодействия клиента и сервера. Опирайтесь на пример технического задания из раздела 5.1.1. определите типы передаваемых сообщений. Например: подключение, готовность, выстрел, пауза, обновление состояния, таблица лидеров и т.д. Для каждого типа оцените частоту отправки и объём данных при передаче в формате JSON. Рассчитайте максимальный трафик при 4 подключённых клиентах. Результат оформите в виде таблицы.

2. Разделение типов сообщений.

Оптимизируйте протокол обмена за счёт разделения сообщений по типам. Вместо универсального JSON-контейнера со всеми полями создайте отдельные структуры для каждого типа запроса и ответа. Например: ShotMessage, ReadyMessage, и т.д. Каждая структура должна содержать только необходимые поля без избыточных данных. Приведите примеры обмена сообщениями «до» и «после» оптимизации. Укажите достигнутое сокращение объёма в байтах для ключевых типов сообщений.

3. Минимизация дублирования и частоты.

Оптимизируйте частоту и содержание отправляемых сообщений. Реализуйте отправку только изменений вместо полного состояния игры. Сократите дублирующиеся данные. Например, передавайте идентификатор игрока вместо полного имени при каждом выстреле. Агрегируйте несколько некритичных событий в одно сообщение. Опишите реализованные механизмы и оцените снижение общего трафика.

4. Управление соединениями и очередями.

Оптимизируйте использование сетевых соединений и обработку сообщений. Используйте пул потоков для обработки входящих сообщений от клиентов. Внедрите механизм приоритетов. Например, обеспечьте обработку вне очереди критических сообщений, таких как выстрел в игре или пауза. Подготовьте краткий отчёт о количестве потоков, размере очередей и времени обработки сообщений «до» и «после» оптимизации.

5.1.4 Типовые задания (оценочное средство - Задания) для оценки сформированности компетенции ППК-Р6:

Задачи для освоения компетенций ППК-Р6 используются из раздела 5.1.1. Компетенции формируются на основании следующих заданий:

1. Система контроля версий и декомпозиция.

Используйте Git на протяжении всей разработки. Выполняйте регулярные коммиты с осмысленными сообщениями. Используйте ветки: main и develop. История коммитов должна наглядно отражать декомпозицию работы на логические этапы. На защите продемонстрируйте историю коммитов и объясните структуру веток.

2. Стиль кода, саморецензирование и модульность.

Ознакомьтесь со стилями кодирования, принятыми в крупных библиотеках с открытым исходным кодом на GitHub, и следуйте выбранному стилю в своём проекте. Проведите саморецензирование кода: найдите и исправьте несколько проблем, связанных со стилем кодирования. Покажите изменения в виде отдельных коммитов. Обеспечьте слабую связанность модулей. Разделите код на модули в соответствии с предметными областями так, чтобы замена одного модуля не требовала переписывания остальных. Например: сеть, бизнес-логика, база данных, пользовательский интерфейс и т.д. Кратко опишите структуру модулей.

3. Конфигурирование и логирование.

Вынесите настройки приложения во внешний конфигурационный файл, чтобы изменение параметров не требовало перекомпиляции. Пример настроек: порт сервера, максимальное количество игроков, параметры подключения к базе данных и т.п. Реализуйте логирование с использованием одной из промышленных библиотек. Примеры библиотек: Log4j или SLF4J+Logback. Логирование должно содержать временную метку, уровень важности (INFO, WARNING, SEVERE) и контекст (например, имя клиента). На защите проекта продемонстрируйте изменение параметров через конфигурационный файл и логирование.

Критерии оценивания (оценочное средство - Задания)

Оценка	Критерии оценивания
зачтено	Выполнена основная часть задания, возможно с незначительными недочетами
не зачтено	Выполнено менее половины задания, есть существенные недочеты

5.2. Описание шкал оценивания результатов обучения по дисциплине при промежуточной аттестации

Шкала оценивания сформированности компетенций

Уровень сформированности компетенций (индикатора достижения компетенций)	плохо	неудовлетворительно	удовлетворительно	хорошо	очень хорошо	отлично	превосходно
	не зачтено			зачтено			
<u>Знания</u>	Отсутствие знаний теоретического материала. Невозможность оценить полноту знаний вследствие отказа обучающегося от ответа	Уровень знаний ниже минимальных требований. Имели место грубые ошибки	Минимально допустимый уровень знаний. Допущено много негрубых ошибок	Уровень знаний в объеме, соответствующем программе подготовки. Допущено несколько негрубых ошибок	Уровень знаний в объеме, соответствующем программе подготовки. Допущено несколько несущественных ошибок	Уровень знаний в объеме, соответствующем программе подготовки. Ошибок нет.	Уровень знаний в объеме, превышающем программу подготовки.
<u>Умения</u>	Отсутствие минимальных умений. Невозможность оценить наличие умений вследствие отказа обучающегося от ответа	При решении стандартных задач не продемонстрированы основные умения. Имели место грубые ошибки	Продемонстрированы основные умения. Решены типовые задачи с негрубыми ошибками. Выполнены все задания, но не в полном объеме	Продемонстрированы все основные умения. Решены все основные задачи с негрубыми ошибками. Выполнены все задания в полном объеме, но некоторые с недочетами	Продемонстрированы все основные умения. Решены все основные задачи. Выполнены все задания в полном объеме, но некоторые с недочетами	Продемонстрированы все основные умения. Решены все основные задачи с отдельными несущественными недочетами, выполнены все задания в полном объеме	Продемонстрированы все основные умения. Решены все основные задачи. Выполнены все задания, в полном объеме без недочетов
<u>Навыки</u>	Отсутствие базовых навыков. Невозможность оценить наличие навыков вследствие	При решении стандартных задач не продемонстрированы базовые навыки. Имели место грубые	Имеется минимальный набор навыков для решения стандартных	Продемонстрированы базовые навыки при решении стандартных задач с	Продемонстрированы базовые навыки при решении стандартных задач без	Продемонстрированы навыки при решении нестандартных задач	Продемонстрирован творческий подход к решению нестандартных задач

	отказа обучающегося от ответа	ошибки	х задач с некоторым и недочетами	некоторым и недочетами	ошибок и недочетов	без ошибок и недочетов	
--	-------------------------------	--------	----------------------------------	------------------------	--------------------	------------------------	--

Шкала оценивания при промежуточной аттестации

Оценка		Уровень подготовки
зачтено	превосходно	Все компетенции (части компетенций), на формирование которых направлена дисциплина, сформированы на уровне не ниже «превосходно», продемонстрированы знания, умения, владения по соответствующим компетенциям на уровне выше предусмотренного программой
	отлично	Все компетенции (части компетенций), на формирование которых направлена дисциплина, сформированы на уровне не ниже «отлично».
	очень хорошо	Все компетенции (части компетенций), на формирование которых направлена дисциплина, сформированы на уровне не ниже «очень хорошо»
	хорошо	Все компетенции (части компетенций), на формирование которых направлена дисциплина, сформированы на уровне не ниже «хорошо».
	удовлетворительно	Все компетенции (части компетенций), на формирование которых направлена дисциплина, сформированы на уровне не ниже «удовлетворительно», при этом хотя бы одна компетенция сформирована на уровне «удовлетворительно»
не зачтено	неудовлетворительно	Хотя бы одна компетенция сформирована на уровне «неудовлетворительно».
	плохо	Хотя бы одна компетенция сформирована на уровне «плохо»

5.3 Типовые контрольные задания или иные материалы, необходимые для оценки результатов обучения на промежуточной аттестации с указанием критериев их оценивания:

5.3.1 Типовые задания (оценочное средство - Контрольные вопросы) для оценки сформированности компетенции ПК-КРМ_3

1. Что включает в себя понятие виртуальная машина?
2. Чем отличается исполнение кода, разработанного на Java и на C?
3. Какие системы сборки Java приложений Вы знаете? Приведите примеры использования
4. Расскажите основные элементы объектно-ориентированного программирования на примере языка программирования Java.
5. Какие SOLID принципы применяются при проектировании Java-приложений? Приведите примеры
6. Какие основные структуры данных реализованы в Java? Приведите примеры
7. Расскажите о лямбда выражениях в Java. Приведите примеры использования.
8. Какие стандартные функциональные интерфейсы реализованы в Java. Приведите примеры использования.
9. Приведите примеры использования Stream API для работы со структурами данных.
10. Что входит в понятие «архитектура приложения»? Какие архитектурные паттерны могут применяться при разработке сетевых приложений.

11. Расскажите о методах реализации шаблона MVC для построения сетевых приложений.
12. Что из себя представляет послойная архитектура? На каких слоях реализуется клиент-серверное взаимодействие. Приведите пример ответственностей слоев с точки зрения сетевого взаимодействия
13. Расскажите о методах разработки многопоточных приложений в Java. Приведите примеры.
14. Какие классы декораторы существуют для работы с входным и выходным потоком данных? Приведите примеры использования.
15. Расскажите о методах сериализации и десериализации данных в Java. (встроенный механизм, JSON, XML).
16. Расскажите основные принципы построения сети. Понятия MAC-адрес, IP-адрес, порт. Принцип передачи данных, допустимые значения с точки зрения разработчика приложений.
17. Расскажите о методах организации клиент-серверного взаимодействия в Java через сокеты.
18. Какие виды сокетов поддерживаются в стандартной библиотеке? Приведите примеры использования
19. Как создать простой сервер на Java с использованием ServerSocket реализующий общение в формате ping-pong?
20. Как обработать несколько клиентов одновременно на сервере в Java? Приведите пример.
21. В чём разница между синхронным и асинхронным взаимодействием в сети? Как это реализовать в Java? Приведите примеры.
22. Как реализовать широковещательную рассылку сообщений в Java используя класс Socket. Приведите реализацию.
23. Как реализовать адресную рассылку сообщений на стороне сервера в Java используя класс Socket. Приведите реализацию.
24. Как обработать разрыв соединения с клиентом на стороне клиента или сервера? Приведите примеры реализации
25. Расскажите о способах построения визуальных приложений на основе Swing и JavaFX.
26. Что такое JDBC? Какие основные классы используются в JDBC для взаимодействия с базами данных? Приведите примеры.
27. Что позволяет делать ORM технология? Приведите пример использования Hibernate.
28. Расскажите о методах построения клиент-серверных приложений с использованием Android SDK и Java SE.
29. Расскажите об основных отличиях в методах разработки мобильных приложений, исполняемых под операционной системой Android.
30. Какие основные отличия существуют между разработкой приложений для операционной системы Android и Windows?

5.3.2 Типовые задания (оценочное средство - Контрольные вопросы) для оценки сформированности компетенции ПК-Опер_1

1. Чем функциональные требования отличаются от нефункциональных? Приведите по одному примеру каждого для вашего варианта.
2. Какое технологическое ограничение в вашем варианте задачи является наиболее жёстким и почему?
3. Почему для клиент-серверной игры выбран текстовый протокол поверх TCP, а не UDP? В каком случае UDP был бы предпочтительнее?
4. Опишите многопоточную модель вашего сервера. Сколько потоков создаётся на одного подключённого клиента? Как синхронизируется доступ к общему состоянию игры?
5. Как в вашем приложении организована рассылка обновлений состояния всем клиентам? В каких случаях можно было бы использовать отправку сообщения конкретному клиенту?
6. Что происходит на сервере при обрыве соединения с клиентом во время игры? Как это влияет на других игроков?

7. Какие сценарии отказоустойчивости вы тестировали, кроме обрыва соединения и паузы? Приведите пример.
8. Какой атрибут качества (логирование, автосохранение, мониторинг) вы добавили в рамках сопровождения? Как он влияет на надёжность системы?
9. Если бы потребовалось масштабировать сервер для 100 одновременных подключений, какой компонент архитектуры стал бы узким местом и как бы вы его переработали?

5.3.3 Типовые задания (оценочное средство - Контрольные вопросы) для оценки сформированности компетенции ПК-ТОП_11

1. Почему при передаче данных в формате JSON важно разделять сообщения по типам, а не использовать универсальный контейнер, включающий все возможные поля?
2. Оцените, во сколько раз можно сократить объём передаваемых сообщений после перехода на отправку только изменений вместо полного состояния игры. Приведите математическое обоснование - формулы и численный пример для вашего варианта.
3. Какие типы сообщений в вашем приложении являются критическими и должны обрабатываться вне очереди? Обоснуйте выбор.
4. Как буферизация отправки сообщений влияет на задержку и пропускную способность? В каких случаях буферизация вредна и от неё лучше отказаться?
5. Что произойдёт с приложением, если 4 клиента начнут одновременно отправлять по 100 сообщений в секунду? Как реализованные вами оптимизации позволят справиться с такой нагрузкой?
6. Почему для идентификации игрока в сообщениях лучше использовать числовой идентификатор, а не строку с именем? Какую экономию трафика это даёт в байтах на одно сообщение?
7. В чём разница между пулом потоков и созданием нового потока на каждого клиента с точки зрения потребления ресурсов?
8. Какой тип сообщений в вашем приложении генерирует наибольший трафик? Как вы можете его оптимизировать?
9. Если бы сетевое соединение имело пропускную способность всего 10 КБ/с, смогла бы ваша программа работать корректно? Почему? Приведите расчёт максимального трафика в вашем приложении.

5.3.4 Типовые задания (оценочное средство - Контрольные вопросы) для оценки сформированности компетенции ППК-Р6

1. Почему в Git для сетевого приложения важно разделять main и develop ветки? Как это помогает, например, при переходе от приложения с локальной логикой к многопоточному взаимодействию клиентов и сервера?
2. Какое сообщение коммита вы бы написали для изменения, которое исправляет ошибку с обрывом соединения при паузе игры? Стоит ли выделять отдельную ветку изменений? Если да, то как ее назвать?
3. Как выбранный вами стиль кода влияет на читаемость сетевой части приложения? Назовите два конкретных правила, которые особенно важны для классов, работающих с сокетами.
4. Как обеспечить слабую связанность между сетевым модулем и модулем бизнес-логики в вашем приложении? Приведите конкретный пример интерфейса или абстракции.
5. Если потребуется заменить библиотеку логирования (например, перейти с Log4j на SLF4J+Logback), какие модули вашего приложения придётся изменить? Почему?
6. Какие параметры сетевого взаимодействия вы вынесли в конфигурационный файл? Почему их важно менять без перекомпиляции?
7. Что произойдёт, если при запуске сервера конфигурационный файл отсутствует или содержит некорректный порт? Как ваше приложение должно реагировать?

8. Приведите пример строки лога для события «клиент с идентификатором 3 отключился по таймауту». Какие уровни важности вы использовали бы для логирования сетевых ошибок и почему?
9. Как логирование на сервере помогает при диагностике проблем с синхронизацией у разных клиентов? Приведите конкретный сценарий.

Критерии оценивания (оценочное средство - Контрольные вопросы)

Оценка	Критерии оценивания
зачтено	Студент ответил на большую часть вопросов возможно с незначительными недочетами.
не зачтено	При ответе студент допускает грубые ошибки в основном материале и решении стандартных задач.

6. Учебно-методическое и информационное обеспечение дисциплины (модуля)

Основная литература:

1. Построение распределенных систем на Java / Свистунов А.Н. - Москва : ИНТУИТ, 2016., <https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=663278&idb=0>.
2. Федоричев Л. А. Реализация многопоточности в языке Java : учебное пособие для вузов / Федоричев Л. А., Букунова О. В.; Букунова О. В. - 2-е изд., стер. - Санкт-Петербург : Лань, 2025. - 72 с. - Книга из коллекции Лань - Информатика. - ISBN 978-5-507-52722-9., <https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=971032&idb=0>.
3. Уорбэртон Р. Лямбда-выражения в Java 8. Функциональное программирование - в массы : монография / Уорбэртон Р. - Москва : ДМК-пресс, 2023. - 194 с. - ISBN 978-5-89818-337-0., <https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=878898&idb=0>.

Дополнительная литература:

1. Основы локальных сетей / Новиков Ю.В., Кондратенко С.В. - Москва : ИНТУИТ, 2016., <https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=663642&idb=0>.
2. Кожомбердиева Г. И. Программирование на языке Java: создание графического интерфейса пользователя : учебное пособие / Кожомбердиева Г. И., Гарина М. И. - Санкт-Петербург : ПГУПС, 2012. - 67 с. - Библиогр.: доступна в карточке книги, на сайте ЭБС Лань. - Книга из коллекции ПГУПС - Информатика. - ISBN 978-5-7641-0402-7., <https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=714733&idb=0>.
3. Клиент-серверное приложение на базе JavaFX / Локтев Д.А. - Москва : МГТУ им. Н.Э. Баумана, 2020., <https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=789504&idb=0>.
4. Бауэр К. Java Persistence API и Hibernate : монография / Бауэр К.; Кинг Г.; Грегори Г. - Москва : ДМК-пресс, 2017. - 632 с. - ISBN 978-5-97060-180-8., <https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=772985&idb=0>.

Программное обеспечение и Интернет-ресурсы (в соответствии с содержанием дисциплины):

1. Виртуальная машина Java - <https://java.com/ru/download/> (Свободное ПО)
2. Официальная документация по языку программирования Java - <https://docs.oracle.com/javase/tutorial/>
3. JDK SDK – средства разработки java программ:
<http://www.oracle.com/technetwork/java/javase/downloads/index.html> (Свободное ПО)
4. IntelliJ IDEA Community Edition – визуальное средство разработки java приложений:
<https://www.jetbrains.com/ru-ru/idea/download/> (Свободное ПО)
5. Android Studio – средство разработки мобильных приложений:
<https://developer.android.com/studio/index.html> (Свободное ПО)

7. Материально-техническое обеспечение дисциплины (модуля)

Учебные аудитории для проведения учебных занятий, предусмотренных образовательной программой, оснащены мультимедийным оборудованием (проектор, экран), техническими средствами обучения, компьютерами.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечены доступом в электронную информационно-образовательную среду.

Программа составлена в соответствии с требованиями ОС ННГУ по направлению подготовки/специальности 01.03.02 - Прикладная математика и информатика.

Автор(ы): Козин Евгений Александрович, кандидат технических наук, доцент.

Заведующий кафедрой: Мееров Иосиф Борисович, кандидат технических наук.

Программа одобрена на заседании методической комиссии от 25.06.2025, протокол № Протокол №11.