МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»
(РУТ (МИИТ)

Рабочая программа дисциплины (модуля),
как компонент образовательной программы
базового высшего образования
по направлению подготовки
09.03.01 Информатика и вычислительная техника,
утвержденной директором ФГАОУ ВО РУТ
(МИИТ) Париновым Д.В.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Асинхронное и параллельное программирование

Направление подготовки: 09.03.01 Информатика и вычислительная техника

Направленность (профиль): Технологии разработки программного обеспечения

Форма обучения: Очная

Рабочая программа дисциплины (модуля) в виде электронного документа выгружена из единой корпоративной информационной системы управления университетом и соответствует оригиналу

Простая электронная подпись, выданная ФГАОУ ВО РУТ (МИИТ)
ID подписи: 937226
Подписал: директор центра Проневич Ольга Борисовна
Дата: 12.11.2026

1. Общие сведения о дисциплине (модуле).

Дисциплина посвящена инженерной разработке конкурентных, асинхронных и параллельных программных систем на платформе Java для корпоративных, транспортных и мобильных цифровых сервисов. Актуальность курса определяется ростом нагрузки на отраслевые информационные системы, переходом к отечественным и свободно распространяемым платформам, развитием цифровых транспортных сервисов и появлением виртуальных потоков в современной Java. В ходе обучения студенты проектируют и реализуют сквозной лабораторный прототип высоконагруженного серверного сервиса, исследуют процессы, потоки, модель памяти Java, синхронизацию, CompletableFuture, ForkJoin, параллельную обработку потоков данных, виртуальные потоки, элементы сопрограмм Kotlin, тестирование и профилирование многопоточного кода.

Целью освоения дисциплины является формирование способности проектировать, реализовывать, тестировать и оценивать производительность асинхронных и параллельных программных решений на платформе Java в условиях разработки надежных корпоративных и транспортно-ориентированных цифровых сервисов.

Для достижения поставленной цели в рамках дисциплины решается комплекс задач, направленных на формирование у обучающихся способности – проектировать архитектуру многопоточных Java-приложений с учетом модели памяти и ограничений платформы, реализовывать асинхронные вычислительные и сетевые сценарии средствами конкурентного программирования Java, применять примитивы синхронизации, потокобезопасные коллекции, виртуальные потоки и механизмы структурирования параллельных задач, проводить модульное, нагрузочное и микробенчмарк-тестирование конкурентного кода, готовить техническую документацию по результатам проектирования и экспериментальной проверки решений.

2. Планируемые результаты обучения по дисциплине (модулю).

Перечень формируемых результатов освоения образовательной программы (компетенций) в результате обучения по дисциплине (модулю):

ИП-2 - Способен оптимизировать производительность программного обеспечения;

ОПК-6 - Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения.

Обучение по дисциплине (модулю) предполагает, что по его результатам обучающийся будет:

Знать:

- концептуальные различия асинхронного, конкурентного, параллельного и реактивного программирования в прикладных Java-системах
- модель процессов, потоков, планирования и переключения контекста в операционных системах и JVM
- жизненный цикл потоков Java и механизмы управления задачами через Thread, Runnable, Callable, Future и ExecutorService
- модель памяти Java, правила happens-before, visibility, ordering, volatile, final и safe publication
- принципы потокобезопасности, гонок данных, атомарности, взаимного исключения, дедлоков, livelock и starvation
- примитивы синхронизации Java, включая synchronized, Lock, ReadWriteLock, Semaphore, CountDownLatch, CyclicBarrier и Phaser
- атомарные классы, compare-and-swap и неблокирующие структуры данных в пакете java.util.concurrent.atomic
- потокобезопасные коллекции Java и критерии выбора ConcurrentHashMap, BlockingQueue, CopyOnWriteArrayList и concurrent-наборов
- архитектуру пулов потоков, очередей задач, политик насыщения, ForkJoinPool и work-stealing-планирования
- модель CompletableFuture, композицию асинхронных стадий, обработку ошибок и управление исполнителями
- Stream API, parallel streams, операции reduce и collect, ограничения побочных эффектов и недетерминированности
- виртуальные потоки Java, их назначение, ограничения, диагностику, правила отказа от пулов виртуальных потоков и управление внешней конкурентностью
- основы структурированной конкурентности и сопоставление виртуальных потоков Java с Kotlin coroutines
- архитектурные модели асинхронных серверных сервисов, включая обработку запроса отдельным потоком, событийный цикл, обратное давление и ограничение ресурсов
- принципы разработки многопоточных сервисов REST на Spring Boot с доступом к PostgreSQL и внешним программным интерфейсам
- методы тестирования конкурентного кода, включая модульные тесты, стресс-тесты, детерминизацию ожиданий и выявление состояний гонки

- методы нагрузочного тестирования и профилирования Java-приложений с использованием JMeter, JFR, VisualVM, JMH и анализа thread dump

- инженерные критерии выбора между синхронным, асинхронным, параллельным и виртуально-поточным решением в серверной и мобильной разработке

- требования к техническому документированию и воспроизводимости экспериментов

- этические и эксплуатационные риски конкурентных программных систем, включая деградацию сервиса, утечки данных, некорректное ограничение нагрузки и непрозрачность диагностики

Уметь:

- уметь проектировать сценарии конкурентной обработки запросов при помощи Java SE concurrency API в условиях ограниченного числа ядер, подключений к БД и внешних сетевых сервисов

- уметь реализовывать многопоточные вычислительные задачи при помощи ExecutorService, Callable, Future и ForkJoinPool в условиях воспроизводимого сравнения задержки и пропускной способности

- уметь реализовывать асинхронные конвейеры обработки данных при помощи CompletableFuture в условиях обработки ошибок, тайм-аутов и отмены задач

- уметь выбирать и применять примитивы синхронизации при помощи synchronized, ReentrantLock, Semaphore, CountDownLatch и Phaser в условиях разделяемого состояния и конкурентного доступа

- уметь проектировать потокобезопасные структуры хранения при помощи java.util.concurrent collections в условиях высокой конкуренции операций чтения и записи

- уметь реализовывать параллельную обработку коллекций при помощи Stream API и parallel streams в условиях отсутствия небезопасных побочных эффектов

- уметь разрабатывать высококонкурентный сервис REST при помощи Spring Boot, Java 21+ и PostgreSQL в условиях обработки запроса отдельным потоком и ограниченных соединений с БД

- уметь применять виртуальные потоки при помощи Executors.newVirtualThreadPerTaskExecutor и Thread.Builder в условиях нагрузки с длительным ожиданием ввода-вывода и запрета на объединение виртуальных потоков в пул

- уметь сопоставлять виртуальные потоки Java и сопрограммы Kotlin при помощи минимальных демонстрационных модулей в условиях мобильного или серверного сценария ожидания внешнего программного интерфейса

- уметь проводить нагрузочные эксперименты при помощи Apache JMeter, JFR и VisualVM в условиях фиксации исходных параметров стенда и анализа деградации пропускной способности

- уметь выполнять микробенчмаркинг конкурентных фрагментов при помощи JMH в условиях отделения прогрева JVM от измеряемой фазы

Владеть:

- навыком разработки и сборки многомодульного Java-проекта с конкурентными компонентами на базе OpenJDK, Gradle или Maven

- навыком реализации синхронизации и безопасного доступа к разделяемому состоянию в прикладном Java-коде

- навыком разработки асинхронного REST-сервиса с PostgreSQL и воспроизводимыми сценариями нагрузки

- навыком применения виртуальных потоков Java для задач с длительным ожиданием ввода-вывода и контролем внешних ограничителей конкурентности

- навыком диагностики race condition, deadlock, thread starvation и деградации производительности по данным тестов, профилировщика и thread dump

- навыком подготовки инженерного портфолио лабораторных работ с исходным кодом, отчетами, результатами измерений и аргументированными выводами

3. Объем дисциплины (модуля).

3.1. Общая трудоемкость дисциплины (модуля).

Общая трудоемкость дисциплины (модуля) составляет 3 з.е. (108 академических часа(ов).

3.2. Объем дисциплины (модуля) в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении учебных занятий:

Тип учебных занятий	Количество часов	
	Всего	Семестр №6
Контактная работа при проведении учебных занятий (всего):	64	64
В том числе:		

Занятия лекционного типа	32	32
Занятия семинарского типа	32	32

3.3. Объем дисциплины (модуля) в форме самостоятельной работы обучающихся, а также в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении промежуточной аттестации составляет 44 академических часа (ов).

3.4. При обучении по индивидуальному учебному плану, в том числе при ускоренном обучении, объем дисциплины (модуля) может быть реализован полностью в форме самостоятельной работы обучающихся, а также в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении промежуточной аттестации.

4. Содержание дисциплины (модуля).

4.1. Занятия лекционного типа.

№ п/п	Тематика лекционных занятий / краткое содержание
1	Инженерный контекст асинхронного и параллельного программирования Рассматриваемые вопросы: - различия асинхронности, конкурентности, параллелизма и реактивности; - роль конкурентного программирования в серверных, мобильных и транспортных цифровых сервисах; - профессиональные требования к надежности, масштабируемости и воспроизводимости инженерных решений; - цикл разработки параллельной программы: декомпозиция, планирование коммуникаций, укрупнение и планирование вычислений; - декомпозиция данных и декомпозиция задач как способы разделения работы между потоками исполнения.
2	Процессы, потоки и планирование выполнения Рассматриваемые вопросы: - модель процесса, потока и контекста выполнения в операционной системе; - планирование, переключение контекста и связь потоков JVM с потоками ОС; - ограничения аппаратных ресурсов при проектировании многопоточных программ.
3	Базовая модель потоков Java Рассматриваемые вопросы: - жизненный цикл Thread и состояния потока Java; - интерфейсы Runnable и Callable, Future и базовые способы получения результата; - обработка interruption, cancellation и ошибок в потоках.
4	Модель памяти Java и видимость данных Рассматриваемые вопросы: - правила happens-before, visibility и ordering;

№ п/п	Тематика лекционных занятий / краткое содержание
	- назначение volatile, final, immutable objects и safe publication; - типовые ошибки публикации разделяемого состояния.
5	Потокобезопасность и дефекты конкурентного кода Рассматриваемые вопросы: - race condition, data race, atomicity violation и check-then-act; - deadlock, livelock, starvation и priority inversion; - инженерные стратегии минимизации разделяемого изменяемого состояния.
6	Примитивы синхронизации Java Рассматриваемые вопросы: - мониторы, synchronized и ReentrantLock; - semaphore, CountDownLatch, CyclicBarrier и Phaser; - Exchanger, сообщения, сигналы и почтовые ящики для обмена данными между потоками; - выбор примитива синхронизации по характеру ресурса и сценарию ожидания.
7	Атомарные операции и неблокирующие подходы Рассматриваемые вопросы: - compare-and-swap и семейство Atomic-классов; - aba-проблема, конкуренция за ресурс и ложное разделение кэша; - условия применимости неблокирующих алгоритмов в Java-приложениях.
8	Потокобезопасные коллекции и очереди Рассматриваемые вопросы: - concurrentHashMap, BlockingQueue и CopyOnWrite-коллекции; - задача производителей и потребителей, обратное давление и ограничение очередей; - критерии выбора коллекции при разных профилях чтения и записи.
9	Пулы потоков и управление задачами Рассматриваемые вопросы: - executorService, ThreadPoolExecutor и политики насыщения; - настройка размера пула для вычислительных задач и задач с ожиданием ввода-вывода; - диагностика очередей задач, голодания потоков и неуправляемого роста задержек.
10	ForkJoin и параллельная декомпозиция вычислений Рассматриваемые вопросы: - work-stealing, recursive task и recursive action; - декомпозиция задач по принципу разделения и объединения результатов; - ограничения ForkJoinPool и риски блокирующих операций.
11	CompletableFuture и асинхронные конвейеры Рассматриваемые вопросы: - композиция thenApply, thenCompose, thenCombine и allOf; - обработка исключений, тайм-аутов и отмены асинхронных стадий; - выбор executor для сетевых, вычислительных и смешанных сценариев.
12	Stream API и parallel streams Рассматриваемые вопросы: - pipeline, intermediate и terminal operations; - reduce, collect, associativity и stateless-функции; - ограничения parallel streams при работе с побочными эффектами и общим ForkJoinPool.
13	Виртуальные потоки Java и Project Loom Рассматриваемые вопросы: - назначение виртуальных потоков и модель обработки запроса отдельным потоком; - закрепление виртуального потока, несущие потоки, локальные данные потока и диагностика; - ограничение конкурентности через семафоры и пулы ресурсов вместо пулов виртуальных потоков.
14	Структурированная конкурентность и сопрограммы Kotlin Рассматриваемые вопросы:

№ п/п	Тематика лекционных занятий / краткое содержание
	- принципы структурированной конкурентности и область жизни параллельных задач; - сопоставление виртуальных потоков Java, CompletableFuture и сопрограмм Kotlin; - сценарии применения сопрограмм в Android и серверной разработке.
15	Асинхронный серверный сервис на Spring Boot Рассматриваемые вопросы: - архитектура сервиса REST с конкурентной обработкой запросов; - взаимодействие с PostgreSQL, внешними программными интерфейсами и ограниченными ресурсами; - устойчивость сервиса при росте нагрузки, тайм-аутах и отказах зависимостей.
16	Тестирование, профилирование и инженерная оценка производительности Рассматриваемые вопросы: - модульные, интеграционные, стрессовые и нагрузочные испытания конкурентного кода; - jmh, JFR, VisualVM, дамп потоков и интерпретация метрик; - оформление воспроизводимого инженерного вывода о корректности и производительности.

4.2. Занятия семинарского типа.

Лабораторные работы

№ п/п	Наименование лабораторных работ / краткое содержание
1	Инициализация проекта с последовательной обработкой данных Студент создает репозиторий, настраивает OpenJDK, Gradle или Maven и JUnit. В проект добавляется синхронный обработчик заявок транспортного сервиса с фиксацией исходных метрик времени выполнения. Результаты запуска сохраняются как исходный уровень для последующих сравнений.
2	Процессы и внешние команды в Java Студент реализует модуль запуска внешних диагностических команд через ProcessBuilder. Код собирает стандартный вывод, поток ошибок, код завершения и время выполнения процесса. Полученные данные включаются в журнал исполнения прототипа.
3	Потоки Java и управляемое завершение задач Студент переносит часть обработки заявок в отдельные потоки Java. В код добавляются сценарии прерывания, корректного завершения и обработки исключений. Поведение потоков проверяется модульными тестами и журналом событий.
4	ExecutorService и пулы потоков для пакетной обработки Студент заменяет ручное создание потоков на ExecutorService и Callable. Для набора заявок измеряются время обработки, размер очереди и число активных задач. Конфигурации пула сравниваются на одинаковом входном наборе.
5	Модель памяти Java и безопасная публикация состояния Студент создает демонстрационный фрагмент с небезопасной публикацией общего состояния и фиксирует нестабильное поведение тестами. Затем код переписывается с immutable-объектами, final-полями или volatile-переменными. В отчете сопоставляются наблюдаемые эффекты до и после исправления.
6	Синхронизация разделяемого ресурса Студент реализует общий счетчик доступных слотов обработки заявок с использованием synchronized и ReentrantLock. Для конкурентного набора операций проверяется отсутствие потерь обновлений. Результаты двух реализаций сравниваются по корректности и времени выполнения.
7	Семафоры, барьеры и координация этапов обработки Студент ограничивает число одновременных обращений к имитированному внешнему API с помощью Semaphore. Для пакетной обработки добавляется координация этапов через

№ п/п	Наименование лабораторных работ / краткое содержание
	CountDownLatch или Phaser. Лог выполнения подтверждает соблюдение ограничений конкурентности.
8	Производители, потребители и потокобезопасные очереди Студент реализует конвейер приема, нормализации и записи заявок через BlockingQueue. Производители и потребители запускаются в разных потоках с контролем завершения. Для очереди задается ограничение размера и проверяется поведение при насыщении.
9	ConcurrentHashMap и конкурентный кэш справочных данных Студент добавляет в прототип потокобезопасный кэш маршрутов или тарифов на базе ConcurrentHashMap. Код выполняет конкурентные чтения и обновления с использованием compute или merge. Тесты проверяют согласованность результатов при одновременных обращениях.
10	ForkJoin для параллельной декомпозиции вычислений Студент реализует вычисление агрегированных показателей по большому набору заявок через ForkJoinPool. Задача декомпозируется на подзадачи с контролем порога разбиения. Производительность сравнивается с последовательной реализацией на одном наборе данных.
11	CompletableFuture для асинхронной агрегации данных Студент реализует асинхронное получение данных из нескольких имитированных источников через CompletableFuture. В цепочку добавляются тайм-ауты, резервные значения и обработка исключений. Итоговый агрегированный ответ используется в прототипе сервиса.
12	Параллельная обработка потоков данных Студент переносит расчет сводных показателей на Stream API и параллельные потоки данных. Операции reduce и collect оформляются без небезопасных побочных эффектов. Полученные результаты сверяются с последовательной версией и фиксируются в отчете.
13	Сервис REST на Spring Boot с конкурентной обработкой запросов Студент разворачивает Spring Boot приложение с конечной точкой REST для обработки заявок. Сервис подключается к PostgreSQL или локальному контейнеру БД и выполняет ограниченный набор операций чтения и записи. Для конечной точки создаются интеграционные тесты.
14	Виртуальные потоки Java для нагрузки с ожиданием ввода-вывода Студент переводит обработку сетевых ожиданий или ожиданий БД на Executors.newVirtualThreadPerTaskExecutor. Ограничение доступа к внешнему ресурсу выполняется через Semaphore или пул соединений. Метрики сравниваются с реализацией на фиксированном пуле платформенных потоков.
15	Сопоставление виртуальных потоков Java и сопрограмм Kotlin Студент создает небольшой Kotlin модуль или демонстрационный фрагмент с сопрограммами для ожидания нескольких внешних операций. Поведение сопоставляется с Java реализацией на виртуальных потоках или CompletableFuture. В портфолио добавляется краткая матрица применимости подходов для серверных и Android сценариев.
16	Нагрузочное тестирование, профилирование и финальная сборка портфолио Студент запускает нагрузочный сценарий Apache JMeter для сервиса REST и собирает данные JFR, VisualVM или дамп потоков. Узкие места классифицируются по загрузке процессора, блокировкам, очередям, БД или внешним ожиданиям. Итоговое портфолио включает исходный код, таблицу метрик, графики и инженерный вывод о выбранной модели конкурентности.

4.3. Самостоятельная работа обучающихся.

№ п/п	Вид самостоятельной работы
1	Изучение рекомендованной литературы.
2	Подготовка к лабораторным работам.

3	Подготовка к промежуточной аттестации.
4	Подготовка к текущему контролю.

5. Перечень изданий, которые рекомендуется использовать при освоении дисциплины (модуля).

№ п/п	Библиографическое описание	Место доступа
1	Баланов, А. Н. Построение микросервисной архитектуры и разработка высоконагруженных приложений : учебное пособие для вузов / А. Н. Баланов. — Санкт-Петербург : Лань, 2024. — 244 с. — ISBN 978-5-507-48747-9. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/394538 (дата обращения: 19.06.2026)
2	Кожомбердиева, Г. И. Программирование на языке Java: многопоточные приложения : учебное пособие / Г. И. Кожомбердиева. — Санкт-Петербург : ПГУПС, 2012. — 44 с. — ISBN 978-7641-0401-0. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/64399 (дата обращения: 19.06.2026)
3	Разработка серверной части web-приложения на базе Spring : методические указания / составители С. А. Коваленко [и др.]. — Воронеж : ВГТУ, 2023. — 35 с. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/383252 (дата обращения: 19.06.2026)
4	Митяков, Е. С. Современная разработка мобильных приложений на языке Kotlin: Практикум : учебное пособие / Е. С. Митяков, М. С. Поздняков. — Москва : РТУ МИРЭА, 2025. — 117 с. — ISBN 978-5-7339-2726-8. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/513041 (дата обращения: 19.06.2026)
5	Наир, В. Предметно-ориентированное проектирование в Enterprise Java : руководство / В. Наир ; перевод с английского А. В. Снастина. — Москва : ДМК Пресс, 2020. — 306 с. — ISBN 978-5-97060-872-2. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/179503 (дата обращения: 19.06.2026)
6	Зарипов, Е. А. Разработка клиент-серверных приложений: Практикум : учебное пособие / Е. А. Зарипов, А. А. Куликов, М. М. Благирев. — Москва : РТУ МИРЭА, 2024 — Часть 2 — 2024. — 80 с. — ISBN 978-5-7339-2269-0. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/448784 (дата обращения: 19.06.2026)
7	Староверова, Н. А. Операционные системы : учебник / Н. А. Староверова. — Санкт-Петербург : Лань, 2022. — 308 с. — ISBN 978-5-8114-4000-9. — Текст : электронный	Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/207089 (дата обращения: 19.06.2026)

6. Перечень современных профессиональных баз данных и информационных справочных систем, которые могут использоваться при освоении дисциплины (модуля).

ЭБС Лань – https://e.lanbook.com/.

Образовательная платформа Юрайт – https://urait.ru/.

Единый реестр российских программ для ЭВМ и баз данных – https://reestr.digital.gov.ru/reestr/.

Профессиональные стандарты и квалификации, справочная информация КонсультантПлюс – https://www.consultant.ru/document/cons_doc_LAW_157436/.

Транспортная стратегия Российской Федерации до 2030 года с прогнозом на период до 2035 года, КонсультантПлюс – https://www.consultant.ru/document/cons_doc_LAW_402052/.

Документация Java SE 21 по java.util.concurrent – https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/concurrent/package-summary.html.

OpenJDK JEP 444 Virtual Threads – https://openjdk.org/jeps/444.

OpenJDK JEP 453 Structured Concurrency – https://openjdk.org/jeps/453.

Документация Kotlin Coroutines – https://kotlinlang.org/docs/coroutines-overview.html.

Документация Spring Framework – https://docs.spring.io/spring-framework/reference/.

Документация Spring Boot – https://docs.spring.io/spring-boot/index.html.

Руководство Gradle – https://docs.gradle.org/current/userguide/userguide.html.

Руководства Apache Maven – https://maven.apache.org/guides/.

Руководство JUnit – https://docs.junit.org/.

Документация PostgreSQL – https://www.postgresql.org/docs/current/.

Руководство Apache JMeter – https://jmeter.apache.org/usermanual/index.html.

Проект OpenJDK JMH – https://openjdk.org/projects/code-tools/jmh/.

Репозиторий OpenJDK jcstress – https://github.com/openjdk/jcstress.

7. Перечень лицензионного и свободно распространяемого программного обеспечения, в том числе отечественного производства, необходимого для освоения дисциплины (модуля).

Операционные системы – Astra Linux, ALT Linux, РЕД ОС, Debian GNU/Linux.

Платформа Java – OpenJDK 21+, Gradle, Apache Maven.

Среды разработки – IntelliJ IDEA Community Edition, Eclipse IDE, Apache NetBeans.

Серверные технологии – Spring Boot, Spring Framework, Spring Web MVC, PostgreSQL, Postgres Pro.

Тестирование и измерения – JUnit, Mockito, Awaitility, Apache JMeter, OpenJDK JMH, OpenJDK jcstress.

Диагностика – JDK Mission Control, Java Flight Recorder, VisualVM, jcmd.

Вспомогательные инструменты – Git, Docker или Podman, curl, Hoppscotch, Kotlin.

8. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю).

Учебные аудитории для проведения учебных занятий, оснащенные компьютерной техникой и наборами демонстрационного оборудования.

Для лабораторных занятий – наличие персональных компьютеров вычислительного класса.

9. Форма промежуточной аттестации:

Экзамен в 6 семестре.

10. Оценочные материалы.

Оценочные материалы, применяемые при проведении промежуточной аттестации, разрабатываются в соответствии с локальным нормативным актом РУТ (МИИТ).

Авторы:

старший преподаватель кафедры
«Цифровые технологии управления
транспортными процессами»
Института железнодорожного
транспорта Е.А. Заманова

Согласовано:

Заведующий кафедрой ЦТУТП В.Е. Нутович

Директор ЦС IT - программ ВИШ О.Б. Проневич

Председатель учебно-методической
комиссии Н.А. Андриянова