

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«Московский физико-технический институт  
(национальный исследовательский университет)»**

**УТВЕРЖДЕНО**

**Директор физтех-школы  
прикладной математики и  
информатики**

**А.М. Райгородский**

	<b>Рабочая программа дисциплины (модуля)</b>
<b>по дисциплине:</b>	Промышленное программирование на языке Java
<b>по направлению:</b>	Прикладная математика и информатика
<b>профиль подготовки:</b>	Проектирование и разработка комплексных бизнес-приложений Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
<b>курс:</b>	2
<b>квалификация:</b>	бакалавр

Семестр, формы промежуточной аттестации: 3 (осенний) - Дифференцированный зачет

Аудиторных часов: 60 всего, в том числе:

лекции: 30 час.

семинары: 30 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 75 час.

Всего часов: 135, всего зач. ед.: 3

Программу составили:

И.Н. Пономарев, канд. физ.-мат. наук, доцент

О.Н. Ивченко, старший преподаватель

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 12.02.2024

## Аннотация

Курс направлен на получение базовых навыков во всех этапах разработки промышленного программного продукта начиная от проектирования архитектуры, написания кода и заканчивая автоматизацией тестирования и развертывания. Основным языком программирования в рамках курса является Java, самый популярный из языков промышленного программирования. Будут рассмотрены основы программирования на Java, средства разработки, сборки и тестирования программ на этом языке. Также в курсе будут затронуты темы паттернов проектирования, test-driven development, continuous delivery/integration, язык графического описания моделей UML.

На практических занятиях будут отрабатываться навыки написания программ на языке Java, связывания программ с другими системами (например, с базами данных и веб-сервисами), тестирования, развертывания, сборки и публикации программ, в том числе в виде веб-сервисов.

## 1. Цели и задачи

### Цель дисциплины

- овладение студентами правил языка программирования Java и приемами использования языка Java в практике программирования.

### Задачи дисциплины

- приобретение студентами навыков проектирования и реализации приложений на языке Java с использованием приемов объектно-ориентированного программирования, примитивов многопоточности и веб-технологий;
- овладение студентами современных практик разработки: использование IDE, системы контроля версий, unit-тестирование.

## 2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
	ОПК-1.3 Способен определять границы применимости полученных результатов
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
	ОПК-2.3 Знает основные требования информационной безопасности

## 3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны знать:

- принцип исполнения программ на Java с использованием JVM;
- принцип работы сборки мусора в Java;
- типы данных языка Java;
- управление потоком выполнения в Java;
- основные классы и возможности стандартной библиотеки;
- правила работы с исключениями;
- принципы разработки параметризованных классов и методов (generics);
- внутреннее строение контейнеров стандартной библиотеки и временную сложность операций с ними;
- потоковая обработка данных при помощи Stream API;
- взаимодействие с реляционными СУБД с помощью JDBC API;
- принципы разработки многопоточного кода в Java и инструменты стандартной библиотеки;
- модель памяти Java;
- возможности Java Reflection API;
- применение аннотаций и обработка аннотаций на уровне Reflection API;
- принцип работы DI-контейнера;
- основные возможности Spring.

уметь:

- реализовывать библиотеку общего назначения на языке Java по заданным интерфейсам;
- добавлять в приложение поддержку многопоточности, анализировать потокобезопасность реализации;
- покрывать код unit-тестами с использованием фреймворка JUnit, анализировать покрытие кода тестами;
- работать с распределенной системой контроля версий git;
- использовать средства code review на сервисе Github;
- реализовывать приложение, предоставляющее HTTP API с помощью фреймворка Spring.

владеть:

- навыками работы с объектами и потоками и кругозором в выборе архитектурного решения поставленной задачи.

#### 4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

##### 4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Обзор истории, возможностей и особенностей экосистемы Java.	5	5		10
2	Примитивные типы. Управление выполнением. Операторы. Массивы.	5	5		10
3	Классы. Интерфейсы. Класс Object и его стандартные методы.	5	5		10
4	Enumerations. Исключения. Строки.	4	3		9
5	JDBC API.	2	3		9
6	Streams API, Optionals.	2	3		9
7	Concurrency.	2	3		9
8	Reflection & DI.	5	3		9
Итого часов		30	30		75
Подготовка к экзамену		0 час.			

## 4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 3 (Осенний)

## 1. Обзор истории, возможностей и особенностей экосистемы Java.

Обзор истории, возможностей и особенностей экосистемы Java. История платформы Java, распространенность языка Java, современные применения платформы и языка Java (desktop, server-side backend, mobile). JVM и предоставляемые ей сервисы: Byte code, интерпретация и JIT-компиляция, Garbage Collection, Threading & Memory Model. Загрузка и исполнение кода в runtime. Рефлексия. Стандартная библиотека. Разнообразие JVM-языков. Критика платформы и языка Java. Ответ на критику: развитие языка, начиная с 8-й версии: lambdas & functional programming, modules, compact strings, type inference. Важнейшие элементы экосистемы Java: Библиотеки и фреймворки: JUnit, JMH, Spring Framework, Lombok, Selenium. Среды разработки: IDE IDEA, Eclipse, NetBeans. Системы сборки: Maven. Gradle Java community: JUGs, Conferences. Oracle certification exams Hello, world: сборка и запуск jar-файла.

## 2. Примитивные типы. Управление выполнением. Операторы. Массивы.

Примитивные типы. Управление выполнением. Операторы. Массивы.

Ключевые слова, идентификаторы и комментарии в Java. Импорт пакетов в java-файле. Примитивные типы данных. Определение переменных и их области видимости. Вывод типов (type inference) при создании переменных (Java 11). Ключевое слово final. Операторы, приоритет, скобки. Конструкции if, if/else и тернарный оператор switch. Массивы: декларирование, инстанцирование, инициализация и использование. Циклы for (две формы). Циклы while и do/while. Использование break и continue. Использование varargs.

## 3. Классы. Интерфейсы. Класс Object и его стандартные методы.

Классы. Интерфейсы. Класс Object и его стандартные методы.

Классы. Структура Java-класса: переменные и методы. Области видимости (private, protected, default, public) классов и элементов класса. Инкапсуляция. Перегрузка (overloading) методов. Конструктор. Конструкторы по умолчанию и явно созданные конструкторы. Секции инициализации. Создание объекта (создание, присвоение ссылки, выход из области видимости, сборка мусора). Жизненный цикл объекта. Наследование. Final-классы. Тип ссылки и тип объекта. Разница между передачей ссылки на объект и передачей примитивного типа в аргумент метода. Переопределение методов. Аннотация @Override. Полиморфизм. Ключевые слова super и this. Абстрактные классы и методы. Приведение типов. Аннотация @SuppressWarnings ("unchecked"). Контракты методов класса Object: equals. Сравнение через == и equals (hashCode). Необходимость согласованного переопределения equals и hashCode. toString finalize. Недостатки идеи делегирования сборщику мусора задачи освобождения системных ресурсов. Вложенные классы. Анонимные классы. Доступ из вложенного/анонимного класса к элементам родительского класса.

Интерфейсы. Множественное наследование с помощью интерфейсов. Ключевое слово instanceof, его работа на классах и интерфейсах. default-методы интерфейсов (Java 8). Функциональные интерфейсы (Java 8). Статические методы и переменные классов и интерфейсов. Статические секции инициализации. Недостатки использования статических методов и "самодельной" реализации паттерна singleton.

## 4. Enumerations. Исключения. Строки.

Enumerations.

Определение. Добавление полей, методов и конструкторов в элементы enum-типов.

Базовые принципы проектирования классов.

Минимизация области видимости. Минимизация мутабельности. Документирование точек расширения через наследование, или запрет на наследование (Effective Java 4.16,17,18).

Исключения.

Создание исключений. Ключевые слова throw и throws. Перехват и обработка исключений. Try-catch, try-multicatch. try-finally. try-with-resources. Интерфейс AutoCloseable. Виды исключений: Checked, Unchecked, Errors. Достоинства и недостатки наличия checked exceptions в языке. Стандартные исключения: NullPointerException, ArithmeticException, ArrayIndexOutOfBoundsException, ClassCastException. Использование InvalidArgumentException и IllegalStateException (Effective Java, 10.72). Причины, по которым не следует использовать исключения для контроля выполнения программы.

Строки.

Пул строк. Конкатенация строк (Effective Java, 63). String.format. StringBuilder. Получение символа на позиции (charAt и codePoints). Полезные и «вредные» методы класса String"

Collections.

Интерфейсы коллекций: Iterable, Collection, Queue, Deque, List, Map, SortedMap, NavigableMap, Set, SortedSet, NavigableSet etc. Оценка сложности алгоритмов. Для чего нужны различные реализации интерфейсов коллекций. Устройство, функциональность и область применимости основных коллекций ArrayList, LinkedList HashMap/HashSet, LinkedHashMap/LinkedHashSet, TreeMap/TreeSet. Итерации по коллекциям: цикл for, метод entrySet(), метод forEach. Интерфейсы java.util.Comparator и java.lang.Comparable. Утилитные классы Collections и Arrays

Lambdas & method references.

Реализация коллбэков через анонимные классы (на примере File.listFiles(...), Java 8 in Action). Необходимость введения лямбд и ссылок на методы. Функциональные интерфейсы. Стандартные функциональные интерфейсы (из пакета java.util.function): Predicate<T>, Function<T,R>, Supplier<T>, Consumer<T>, UnaryOperator<T>, BinaryOperator<T> и т. п. Совместимость типов функциональных интерфейсов, лямбд и анонимных классов. Вывод типов параметров лямбд. Композиция функциональных интерфейсов: reversed & thenComparing для Comparator<>, compose для Function<>, and/or/etc.. для Predicate<> и т. п.

## 5. JDBC API.

JDBC API JDBC Drivers Connection, Statement, ResultSet ConnectionPool SQL Injection.

## 6. Streams API, Optionals.

Streams API.

Что такое поток? Коллекции vs потоки. Внутренние vs внешние итерации Stream pipeline. Промежуточные vs. терминальные операции. Промежуточные операции: filter(), map(), peek(). Версии map для примитивных типов. Метод flatMap(). Поиск элементов по findFirst, findAny, anyMatch, allMatch, noneMatch distinct. Терминальные операции max, min, countreduce collect. Коллекторы.

Optionals.

Класс Optional<T>: “вырожденный случай” потока. Методы Optional. Корректное использование Optional: как возвращаемый результат, но не как параметр метода. Best practices в использовании Stream API и Optionals.

## 7. Concurrency.

Concurrency-1.

Теоретические ограничения параллелизма: Закон Амдала и USL.

Запуск параллельного выполнения: интерфейсы Runnable, Callable, Future<T>, ExecutorService.submit(). Параллельный доступ к данным. Mutual exclusion vs. visibility. Ключевые слова synchronized и volatile. Happens-before guarantee. Синхронизация shared state. Synchronized-блок, минимизация синхронизации. (Effective Java 78, 79). Преимущества использования ExecutorService по сравнению с низкоуровневыми примитивами wait/notify. Методы invokeAll, invokeAny. ForkJoinPool и Parallel Streams. (Effective Java 81).

Concurrency-2. Аннотации. Reflection API.

Concurrency (окончание).

Потокобезопасные коллекции. Синхронизированные коллекции, CopyOnWriteArrayList, ConcurrentLinkedQueue, ConcurrentHashMap, ConcurrentSkipListMap. Atomic-типы данных. CAS-операции. Неблокирующие алгоритмы. Непрокирующая синхронизация shared state. Механизм кооперативного прерывания CompletableFuture.

Аннотации.

Мотивация создания аннотаций в коде. Синтаксис определения аннотаций. Стандартные аннотации.

Reflection API.

Класс Class. Загрузка ресурсных файлов.

## 8. Reflection & DI.

Reflection & DI.

Reflection API (окончание). Слоистая архитектура приложения. Singleton (anti)pattern. Dependency Injection. Принцип работы DI-контейнера.

Spring-1.

Принцип работы DI-контейнера (окончание). Spring Framework: Spring DI, Spring AOP.

Spring-2.

Spring AOP (окончание). Spring Boot Тестирование Spring Boot приложений.

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

Учебная аудитория, оснащенная компьютером и мультимедийным оборудованием (проектор, звуковая система), а так же персональными компьютерами для обучающихся.

## 6.Перечень рекомендуемой литературы

Основная литература

1. Программирование на Java [Текст] : курс лекций для студентов вузов / Н. А. Вязовик .— М. : Интернет-Ун-т Информ. технологий, 2003 .— 592 с.
2. Разработка приложений Java EE 6 в NetBeans 7 [Электронный ресурс] / Д. Хеффельфингер . — М., ДМК Пресс, 2013.— URL: <https://e.lanbook.com/book/58693> (дата обращения: 25.01.2021). - Полный текст (Режим доступа : из сети МФТИ / Удаленный доступ)

Дополнительная литература

## 7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

Не используются

## 8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

Oracle Java JDK последней версии.

Редактор IntelliJIDEA либо Eclipse последней версии.

## 9. Методические указания для обучающихся по освоению дисциплины (модуля)

Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Самостоятельная работа включает в себя:

- проработку учебного материала (по конспектам лекций, учебной и научной литературе), подготовку ответов на вопросы, предназначенных для самостоятельного изучения, доказательство отдельных утверждений, свойств;
- подготовку к практическим занятиям, выполнение двух индивидуальных домашних заданий.

Промежуточный контроль знаний проводится в виде письменных опросов по теории, а также студенту в ходе освоения курса необходимо выполнить две домашние индивидуальные работы с их последующей защитой.

**ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)**

<b>по направлению:</b>	Прикладная математика и информатика
<b>профиль подготовки:</b>	Проектирование и разработка комплексных бизнес-приложений Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
<b>курс:</b>	<u>2</u>
<b>квалификация:</b>	бакалавр

Семестр, формы промежуточной аттестации: 3 (осенний) - Дифференцированный зачет

**Разработчики:**

И.Н. Пономарев, канд. физ.-мат. наук, доцент

О.Н. Ивченко, старший преподаватель



## 1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
	ОПК-1.3 Способен определять границы применимости полученных результатов
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
	ОПК-2.3 Знает основные требования информационной безопасности

## 2. Показатели оценивания компетенций

В результате изучения дисциплины «Промышленное программирование на языке Java» обучающийся должен:

### знать:

- принцип исполнения программ на Java с использованием JVM;
- принцип работы сборки мусора в Java;
- типы данных языка Java;
- управление потоком выполнения в Java;
- основные классы и возможности стандартной библиотеки;
- правила работы с исключениями;
- принципы разработки параметризованных классов и методов (generics);
- внутреннее строение контейнеров стандартной библиотеки и временную сложность операций с ними;
- потоковая обработка данных при помощи Stream API;
- взаимодействие с реляционными СУБД с помощью JDBC API;
- принципы разработки многопоточного кода в Java и инструменты стандартной библиотеки;
- модель памяти Java;
- возможности Java Reflection API;
- применение аннотаций и обработка аннотаций на уровне Reflection API;
- принцип работы DI-контейнера;
- основные возможности Spring.

### уметь:

- реализовывать библиотеку общего назначения на языке Java по заданным интерфейсам;
- добавлять в приложение поддержку многопоточности, анализировать потокобезопасность реализации;
- покрывать код unit-тестами с использованием фреймворка JUnit, анализировать покрытие кода тестами;
- работать с распределенной системой контроля версий git;
- использовать средства code review на сервисе Github;
- реализовывать приложение, предоставляющее HTTP API с помощью фреймворка Spring.

### владеть:

- навыками работы с объектами и потоками и кругозором в выборе архитектурного решения поставленной задачи.

## 3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

1. Java, распространенность языка Java, современные применения платформы и языка Java.
2. Импорт пакетов в java-файле.
3. Примитивные типы данных.
4. Класс Object и его стандартные методы.
5. Конструкторы по умолчанию и явно созданные конструкторы.
6. Секции инициализации.
7. Множественное наследование с помощью интерфейсов.
8. Статические секции инициализации.
9. Базовые принципы проектирования классов.
10. Минимизация области видимости.
11. Для чего нужны различные реализации интерфейсов коллекций.
12. Стандартные функциональные интерфейсы.

#### **4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся**

1. Чем отличается переопределение от перегрузки?
2. Чем отличается абстрактный класс от интерфейса? Укажите, как изменились эти отличия в Java 8+.
3. Какими способами можно создать Stream в Java.
4. Чем различаются JRE, JVM и JDK?
5. О чем говорит ключевое слово final?
6. Какими значениями инициализируются переменные по умолчанию?
7. Для чего используется модификатор abstract?
8. Что такое маркерные интерфейсы? Чем они отличаются от обычных?
9. Для чего в Java используются статические блоки инициализации?
10. Может ли статический метод быть переопределён или перегружен?

##### **Пример билета**

###### **Билет №1**

1. Для чего в Java используются статические блоки инициализации?
2. Может ли статический метод быть переопределён или перегружен?

##### **Критерии оценивания**

отлично (10) - выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений;

отлично (9) - выставляется студенту, показавшему свободное оперирование знаниями учебной программы дисциплины, выполнение заданий творческого характера;

отлично (8) - выставляется студенту, показавшему владение программным учебным материалом с наличием несущественных ошибок в действиях, самостоятельно исправляемых учащимся;

хорошо (7) - выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности;

хорошо (6) - выставляется студенту если он осознает воспроизведение программного учебного материала, в том числе и различной степени сложности, с несущественными ошибками, затруднения в применении отдельных навыков;

хорошо (5) - выставляется студенту если теоретическое содержание освоено не полностью, некоторые практические навыки сформированы недостаточно, в некоторых случаях были допущены ошибки;

удовлетворительно (4) - выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации;

удовлетворительно (3) - выставляется студенту в случае большого количества недочетов и неправильных ответов, а также пассивной работе в ходе занятий, многие учебные задания не выполнены;

неудовлетворительно (2) - выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных понятий дисциплины и не умеет использовать полученные знания при решении типовых практических задач;

неудовлетворительно (1) - выставляется студенту, который не освоил теоретическое и практическое содержание курса, все выполненные учебные задания содержат грубые ошибки.

## **5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности**

Дифференцированный зачет проводится в устной форме. Во время проведения обучающиеся могут пользоваться программой дисциплины.