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

Рабочая программа дисциплины (модуля)

по дисциплине:	Программирование на языке Java
по направлению:	Прикладная математика и информатика
профиль подготовки:	Искусственный интеллект и большие данные Сетевое обучение кафедра алгоритмов и технологий программирования
курс:	2
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 4 (весенний) - Экзамен

Аудиторных часов: 60 всего, в том числе:
лекции: 30 час.
семинары: 30 час.
лабораторные занятия: 0 час.
Самостоятельная работа: 54 час.

Подготовка к экзамену: 30 час.

Всего часов: 144, всего зач. ед.: 4

Программу составили: И.Н. Пономарев, канд. физ.-мат. наук, доцент
О.Н. Ивченко, старший преподаватель

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 30.08.2022

Аннотация

Целью данного курса является 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, развёртывания, 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-2 Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-2.1 Использует знания основных положений и концепций в области программирования, архитектуру языков программирования, основную терминологию и базовые алгоритмы, основные требования информационной безопасности для практического применения
	ОПК-2.2 Анализирует типовые языки программирования, составляет программы
	ОПК-2.3 Применяет на практике опыт решения задач с использованием базовых алгоритмов, анализа типов коммуникаций и интеграции различных типов программного обеспечения
ОПК-3 Способен использовать и адаптировать существующие математические методы и системы программирования для разработки и реализации алгоритмов решения прикладных задач	ОПК-3.1 Применяет знания основных математических методов и владеет навыками их адаптации для решения конкретной прикладной задачи
	ОПК-3.2 Разрабатывает и реализует алгоритм решения прикладной задачи
	ОПК-3.3 Демонстрирует практический опыт решения прикладных задач с использованием

	систем программирования и специализированного программного обеспечения

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	2	2		6
2	Примитивные типы. Управление выполнением. Операторы. Массивы.	2	2		6
3	Классы. Интерфейсы. Класс Object и его стандартные методы.	4	4		7
4	Enumerations. Исключения. Строки	6	6		7
5	JDBC API	2	2		7
6	Streams API, Optionals	2	2		7
7	Concurrency	5	5		7
8	Reflection & DI	7	7		7

Итого часов	30	30		54
Подготовка к экзамену	30 час.			
Общая трудоёмкость	144 час., 4 зач.ед.			

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

Семестр: 4 (Весенний)

1. Обзор истории, возможностей и особенностей экосистемы Java

Лекция 1. 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-языковКритика платформы и языка Java. Ответ на критику: развитие языка, начиная с 8й версии: lamdbas & 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.

Лекция 2 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 finalОператоры, 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.

Лекция 3. 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
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

Лекция 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 throwsПерехват 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 InvalidStateException (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"

Лекция 7. Collections, Lambdas & Method References

Collections

Интерфейсы коллекций: Iterable, Collection, Queue, Deque, List, Map, SortedMap, NavigableMap, Set, SortedSet, NavigableSet etc. O-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 интерефейсы. 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

Лекция 8. JDBC API

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

6. Streams API, Optionals

Лекция 9. 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

Лекция 10. 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)"
Лекция 11. Concurrency-2. Аннотации. Reflection API
Concurrency (окончание)
Потокобезопасные коллекции. Синхронизированные коллекции,CopyOnWriteArrayList, ConcurrentLinkedQueue,ConcurrentHashMap, ConcurrentSkipListMap. Atomic-типы данных. CAS-операции. Неблокирующие алгоритмы. Неблокирующая 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

Лекция 12. Reflection & DI
Reflection API (окончание). Слоистая архитектура приложения. Singleton (anti)pattern. Dependency Injection. Принцип работы DI-контейнера."
Лекция 13. Spring-1
Принцип работы DI-контейнера (окончание). Spring Framework: Spring DI, Spring AOP"
Лекция 14. Spring-2
Spring AOP (окончание). Spring Boot Тестирование Spring Boot приложений

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

аудитория с доской и проектором/плазменной панелью.

6.Перечень рекомендуемой литературы

Основная литература
Дополнительная литература

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

http://docs.oracle.com/javase/specs/jls/se8/html/index.html - The Java Language Specification.
https://google-styleguide.googlecode.com/svn/trunk/javaguide.html — Google Java Style Guide.

литература:
1. Блох Джошуа: Java. Эффективное программирование — 3е изд. — М.: Вильямс, 2018
2. Гетц Брайан и др.: Java Concurrency на практике — СПб.: Питер, 2020
3. Урма Рауль-Габриель и др.: Современный язык Java. Лямбда-выражения, потоки и функциональное программирование — СПб.: Питер, 2020
4. Библиотека профессионала. Java 2 [Текст] : Т. 1. Основы/К. С. Хорстманн, Г. Корнелл , Java 2. Vol. 1 : Fundamentals, [пер. с англ. Н. А. Мухина], -М., И. Д. Вильямс, 2013
5. Библиотека профессионала. Java 2 [Текст] : Т. 2. Тонкости программирования/К. С. Хорстманн, Г. Корнелл , [пер. с англ. Я. П. Волковой, Д. Я. Ивченко, под ред. Ю. Н. Артеменко], Java 2. Vol. 2 : Advanced Features, -М., Вильямс, 2009

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:

ПРИЛОЖЕНИЕ

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Прикладная математика и информатика
профиль подготовки: Искусственный интеллект и большие данные
Сетевое обучение
кафедра алгоритмов и технологий программирования
курс: 2
квалификация: бакалавр

Семестр, формы промежуточной аттестации: 4 (весенний) - Экзамен

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-2 Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-2.1 Использует знания основных положений и концепций в области программирования, архитектуру языков программирования, основную терминологию и базовые алгоритмы, основные требования информационной безопасности для практического применения
	ОПК-2.2 Анализирует типовые языки программирования, составляет программы
	ОПК-2.3 Применяет на практике опыт решения задач с использованием базовых алгоритмов, анализа типов коммуникаций и интеграции различных типов программного обеспечения
ОПК-3 Способен использовать и адаптировать существующие математические методы и системы программирования для разработки и реализации алгоритмов решения прикладных задач	ОПК-3.1 Применяет знания основных математических методов и владеет навыками их адаптации для решения конкретной прикладной задачи
	ОПК-3.2 Разрабатывает и реализует алгоритм решения прикладной задачи
	ОПК-3.3 Демонстрирует практический опыт решения прикладных задач с использованием систем программирования и специализированного программного обеспечения

2. Показатели оценивания компетенций

В результате изучения дисциплины «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. Могут ли быть final:

- классы,

- интерфейсы,

- методы,

- переменные?

Для тех случаев, где это возможно, опишите к чему это приведёт.

2. Могут ли быть статическими:

- классы,

- интерфейсы,

- методы,

- переменные?

Для тех случаев, где это возможно, опишите к чему это приведёт.

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? Укажете, 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

1.Чем различаются JRE, JVM и JDK?

2. Для чего в Java используются статические блоки инициализации?

Критерии оценивания

отлично

10 всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений;

9 систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, правильное обоснование принятых решений;

8 глубокие знания учебной программы дисциплины и умение применять их на практике при решении конкретных задач, правильное обоснование принятых решений;

хорошо

7 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;

6 знает материал, грамотно излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности;

5 знает основной материал, грамотно излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач неточности;

удовлетворительно

4 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;

3 характер знаний достаточен для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации;

неудовлетворительно

2 не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных понятий дисциплины и не умеет правильно использовать полученные знания при решении типовых практических задач.

1 не знает формулировок основных понятий дисциплины и не умеет использовать полученные знания при решении типовых практических задач.

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

Экзамен проводятся в устной форме. Во время проведения обучающиеся могут пользоваться программой дисциплины.