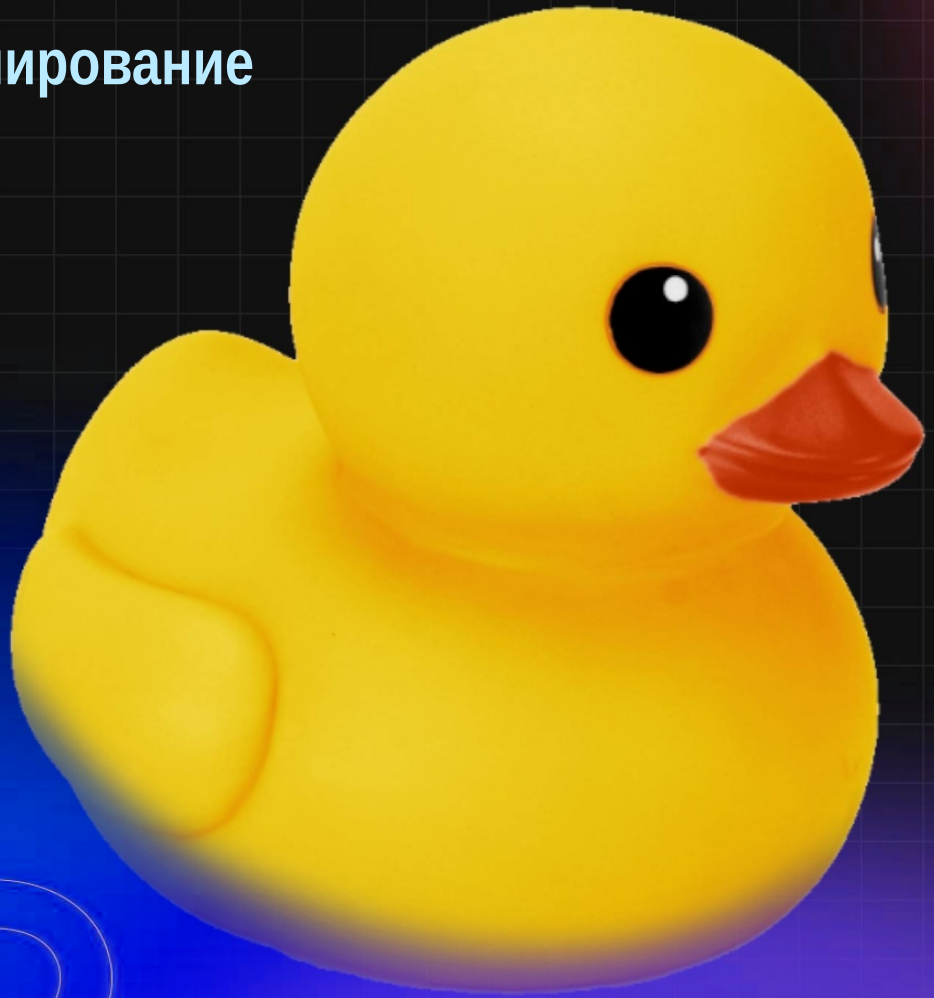
Программирование
2 семестр
2025



Рефлексия

Рефлексия

ИТМО

- Информация в переменных класса и объекта — данные.
- Информация о структуре класса — метаданные.
- Рефлексия - механизм работы с метаданными, позволяющий во время выполнения программы получать информацию о классах, методах, полях, и управлять ими.
- Класс java.lang.Class
- Пакет java.lang.reflect.*

Java

Применение рефлексии

ИТМО

- Фреймворки и библиотеки
  - Dependency Injection (DI)
  - Inversion of Control (IoC)
  - Object-Relationship Mapping (ORM)
- Сериализация и десериализация XML/JSON
- Инструменты для тестирования (JUnit).
- Динамические прокси
- Отладка и анализ кода во время выполнения.

java.lang.Class

ИТМО

- Класс Class<T> - класс, представляющий классы
- Как получить объект класса Class?
  - obj.getClass() — по имеющемуся объекту
  - Class.forName(String name) — динамически по имени
  - T.class — по любому имеющемуся типу данных
    - даже примитивному - int.class
  - Integer.TYPE
  - ClassLoader::defineClass
  - java.lang.invoke.MethodHandler.Lookup::defineClass

Загрузка классов

ИТМО

```
public class A {
  public static void main(String... args) {
    B b = new B();
    System.out.println(b.getName());
  }
}
```

- java.lang.ClassLoader

```
A.class.getClassLoader().loadClass("B");
```

Java

Загрузка классов

ИТМО

```
public class A {
  public static void main(String... args) {
    B b = new B();
    System.out.println(b.getName());
  }
}
```

- java.lang.ClassLoader

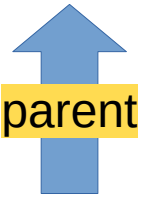
```
A.class.getClassLoader().loadClass("B");
```

Java

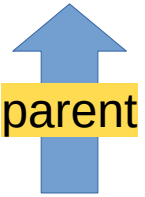
Иерархия загрузчиков

ИТМО

- Загрузчик базовых классов (Bootstrap ClassLoader)
  - встроен в JVM
  - загружает самые основные классы



- Загрузчик классов платформы - getPlatformClassLoader()



- Загрузчик прикладных классов - getSystemClassLoader()
- и т.д.

Java

Загрузка классов

ИТМО

- Процесс загрузки класса
  - Загрузка байт-кода
  - Компоновка (linking)
    - Верификация байт-кода
    - Выделение памяти, разрешение зависимостей
      - загрузка связанных классов
    - Инициализация (начиная со статических блоков)

Java

Информация о классе

ИТМО

- isInterface()
- isArray()
- isPrimitive()
- isEnum()
- isRecord()
- isAnnotation()
- isMemberClass()
- isLocalClass()
- isAnonymousClass()
- isSealed()
- isSynthetic()
- isHidden()

Java

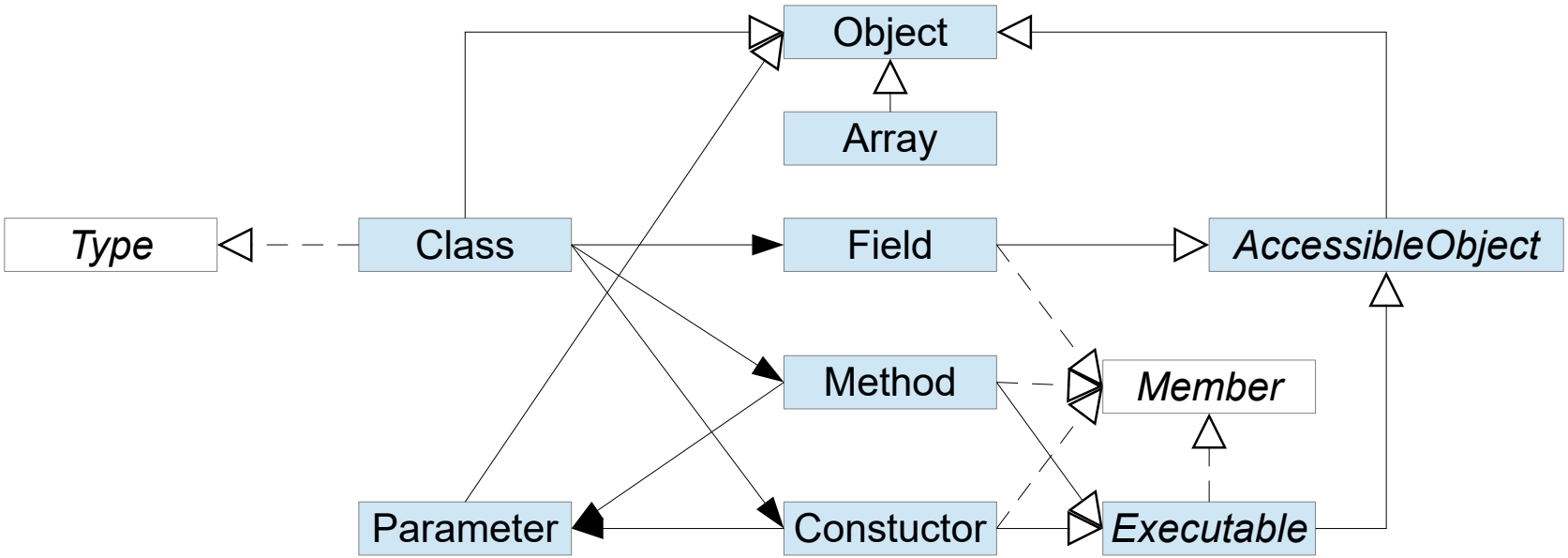
Информация о классе

ИТМО

- Class getSuperclass()
- Class getEnclosingClass()
- Class getDeclaringClass() - null для анонимных классов
- Class[] getInterfaces()
- int getModifiers()
- TypeVariable[] getTypeParameters()

Java

пакет java.lang.reflect



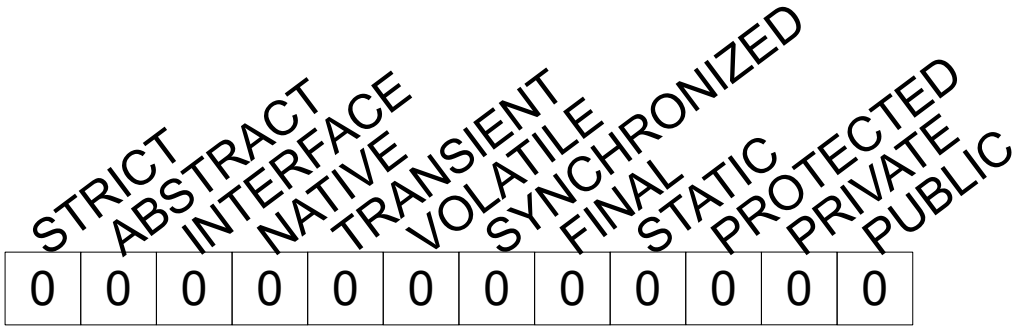
java.lang.reflect.Field

ИТМО

- Object get(Object o)
- int getInt(Object o) // Long, Char, Double, ...
- void set(Object o, Object value)
- void setInt(Object o, int value) // Long, Char, Double, ...
- String getName()
- Class getType() / getGenericType()
- int getModifiers()
- setAccessible(true)

Java

java.lang.reflect.Modifier



- Modifier.PUBLIC ...
- boolean Modifier.isPublic() ...
- int Modifier.fieldModifiers() ...

```
if (field.getModifiers() &
(Modifier.PUBLIC | Modifier.STATIC | Modifier.FINAL) != 0)
```

java.lang.reflect.Array

ИТМО

- newInstance(class, length)
- Object get(Object array, int index)
- int getInt(Object array, int index) // Long, Char, Double, ...
- void set(Object array, int index, Object value)
- void setInt(Object array, int index, int value) // Long, Char, ...
- String getName()
- Class getComponentType() (метод класса Class)

Java

Тип элемента массива

ИТМО

- boolean [Z
- byte [B
- char [C
- class or interface [L*name*;
- double [D
- float [F
- int [I
- long [J
- short [S

java.lang.reflect.Method

ИТМО

- Object invoke(Object o, Object... args)
- String getName()
- Class getReturnType() / getGenericReturnType()
- Class[] getParameterTypes() / getGenericParameterTypes()
- Class[] getExceptionTypes() / getGenericExceptionTypes()
- int getModifiers()

Java

java.lang.reflect.Constuctor

- T newInstance(Object... args)
- String getName()
- Class[] getParameterTypes() / getGenericParameterTypes()
- Class[] getExceptionTypes() / getGenericExceptionTypes()
- int getModifiers()

Field / Method / Constructor

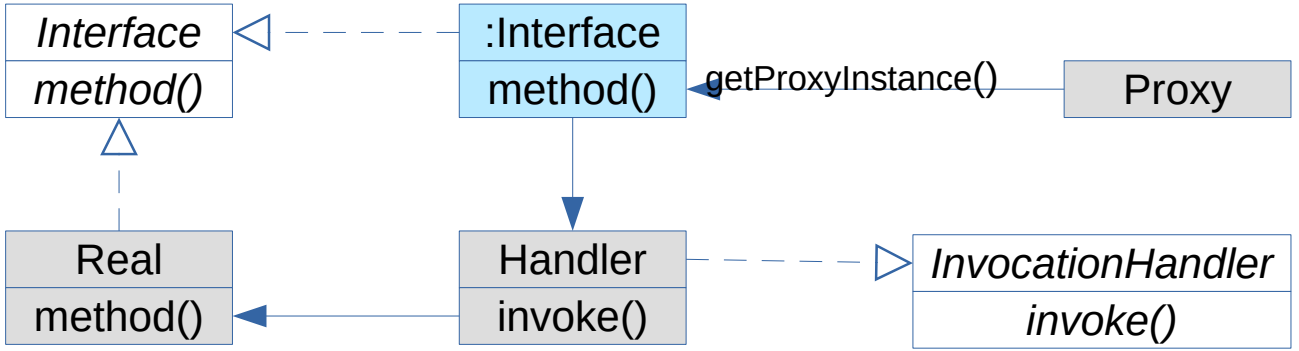
ИТМО

- Возвращают наследуемые / Не возвращают приватные
  - getField(name) / getFields()
  - getMethod(name) / getMethods()
  - getConstructor(name) / getConstructors()
- Возвращают приватные / Не возвращают наследуемые
  - getDeclaredField(name) / getDeclaredFields()
  - getDeclaredMethod(name) / getDeclaredMethods()
  - getDeclaredConstructor(name) / getDeclaredConstructors()

Java

Динамические прокси

- класс java.lang.reflect.Proxy
- интерфейс java.lang.reflect.InvocationHandler
- Создает динамический прокси-объект, реализующий заданные интерфейсы
- Вызовы методов интерфейсов передаются обработчику, реализующему InvocationHandler



Пример кода

ИТМО

```
interface Able {
  public void doIt();
}
```

```
public class Real implements Able {
  public void doIt() { ... }
}
```

```
Real real = new Real();

InvocationHandler handler = (proxy, method, args) -> {
  // ...
  Object result = method.invoke(real, args);
  // ...
  return result;
};

Able proxy = (Able) Proxy.newProxyInstance(
  Able.class.getClassLoader(),
  new Class<?>[] { Able.class },
  handler);

proxy.doIt();
```

Применение рефлексии

- Расширяемость приложений
- Доступ к внутренним элементам класса
- Среды визуальной разработки
- Отладчики
- Библиотеки для тестов

Недостатки рефлексии

ИТМО

- Снижение производительности
- Проблемы с безопасностью
- Раскрытие приватных элементов
- Элементы и аргументы представлены в виде Object[]
- Перегрузка методов - одно имя, разные сигнатуры
- Плохая поддержка упаковки и распаковки

Java

Новые механизмы выполнения методов

ИТМО

- java.lang.invoke.MethodHandler
  - класс для поиска и вызова методов
  - производительность выше, чем у базовой рефлексии
  - "ссылка" на код - абстракция вызова
  - java.lang.reflect.Method - абстракция объявления
- Получить фабрику поиска (MethodHandles.Lookup)
- Задать сигнатуру (MethodType)
- Получить через Lookup ссылку на метод (MethodHandle)
- Вызвать метод через объект-ссылку

MethodHandle.Lookup

ИТМО

- Получаем фабрику для дальнейшего поиска
- Статические методы в классе MethodHandles

```
var lookup = MethodHandles.publicLookup();

var lookup = MethodHandles.lookup();
```

Java

MethodType

ИТМО

- Задаем тип результата и типы аргументов
- Класс MethodType

```
var mt = MethodType.methodType(List.class, Object[].class);

var mt = MethodType.methodType(int.class, Object.class);
```

Java

MethodHandle

ИТМО

- Находим нужный метод с помощью фабрики
- Класс MethodHandle
  - findVirtual(), findStatic(), findConstructor(), **unreflect()**
  - findGetter(Class where, String name, Class field), findSetter()

```
var mt = MethodType.methodType(String.class, String.class);
var mh = lookup.findVirtual(String.class, "concat", mt);
Method prv = Hello.class.getDeclaredMethod("privateHello");
prv.setAccessible(true);
var mh = lookup.unreflect(prv);
```

invoke

ИТМО

- Вызываем метод косвенно через ссылку
- invokeExact() - фиксированное число и тип аргументов
- invoke() - конверсия типов
- invokeWithArguments() - конверсия типов, переменное число аргументов

```
var mt = MethodType.methodType(String.class, String.class);
var mh = lookup.findVirtual(String.class, "concat", mt);
String result = mh.invoke("hello ", "world");
```

Java

Пример

ИТМО

```
import java.lang.invoke.*;
public class MHHello {
  public void hello(String s) {
    System.out.println("Hello, " + s);
  }
  public static void main(String... args) {
    var lookup = MethodHandles.publicLookup();
    var mtype = MethodType.methodType(void.class, String.class);
    var mh = lookup.findVirtual(MHHello.class, "hello", mtype);
    var obj = new MHHello();
    mh.invoke(obj, "world");
  }
}
```

```
obj.hello("world");
```

Вызов методов в JVM до Java 7

ИТМО

- invokestatic
  - статическая диспетчеризация (по типу ссылки)
- invokevirtual
  - динамическая диспетчеризация (по типу объекта)
- invokeinterface
  - динамическая диспетчеризация (много интерфейсов)
- invokespecial
  - статическая диспетчеризация (конструкторы, private, super.)

Java

invokedynamic (indy)

ИТМО

- Динамическое связывание метода
  - 1-е выполнение invokedynamic в точке вызова
    - JVM находит bootstrap-метод (BSM) в пуле констант класса
    - Вызов BSM (name, type, args, ...), который подбирает метод
    - BSM возвращает CallSite с MethodHandle внутри
    - JVM связывает invokedynamic с CallSite и выполняет invoke
  - 2-е выполнение invokedymamic
    - JVM вызывает invoke из MethodHandle связанного CallSite.

Java

Новые механизмы доступа к переменным

ИТМО

- java.lang.invoke.VarHandler
  - типобезопасная ссылка на поле или элемент массива
  - поддержка атомарных операций и модели памяти
  - производительность выше, чем у базовой рефлексии
- Получить фабрику поиска (MethodHandles.Lookup)
- Получить через Lookup ссылку на поле (findVarHandle) или элемент массива (arrayElementVarHandle)
- Обратиться к переменной через объект-ссылку

Режимы доступа к памяти

ИТМО

- VarHandle поддерживает:
- Атомарные операции - compareAndSet, getAndAdd, ...
- Режимы доступа к памяти
  - Plain - без гарантий, самый быстрый
  - Opaque - порядок операций в одном потоке
  - Acquire - операции **до** чтения не будут выполняться **после**
  - Release - операции **после** записи не будут выполняться **до**
  - Volatile - порядок всех операций между потоками

Java

Аннотации

ИТМО

- Аннотации — метаданные, добавляемые в исходный код, не влияющие семантически на программу, но используемые в процессе анализа кода, компиляции или во время исполнения
- Аннотация — это подвид интерфейса, все аннотации — потомки интерфейса java.lang.annotation.Annotation
- Обозначаются символом @

Java

Стандартные аннотации

ИТМО

- @Deprecated — отмечает устаревший метод
- @Override — отмечает переопределенный метод
- @SuppressWarnings — запрещает компилятору выдавать определенные предупреждения
- @FunctionalInterface — показывает, что объявленный тип является функциональным интерфейсом
- @SafeVarargs — не выводить предупреждения об использовании нематериализуемого типа (non-reifiable) в методе или конструкторе с переменным числом аргументов (varargs)

  m(T[]...)

Стандартные мета-аннотации

ИТМО

- @Retention(RetentionPolicy.SOURCE, CLASS, RUNTIME) — время действия аннотации
- @Target(ElementType.FIELD, METHOD, TYPE, CONSTRUCTOR, PACKAGE, LOCAL_VARIABLE, PARAMETER, ANNOTATION_TYPE) — элемент, к которому можно применить аннотацию
- @Inherited — аннотация наследуется потомками
- @Documented — аннотированный элемент должен быть документирован с помощью javadoc
- @Repeatale — аннотация может повторяться для одного и того же элемента

Создание собственных аннотаций

ИТМО

```
@Retenion(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@interface DBTable {
    String name();
}
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
@interface PrimaryKey { }

@DBTable(name="person")
public class Human {
    @PrimaryKey int id;
    String name;
    LocalDate birthdate;
}
```

Аннотации во время выполнения

ИТМО

- интерфейс java.lang.reflect.AnnotatedElement
- реализуется классами Class, Field, Method, Constructor, Parameter
- Annotation[] getAnnotations() / getDeclaredAnnotations()
- Annotation getAnnotation(Class a) / getDeclaredAnnotation
- boolean isAnnotationPresent(Class a)

Java

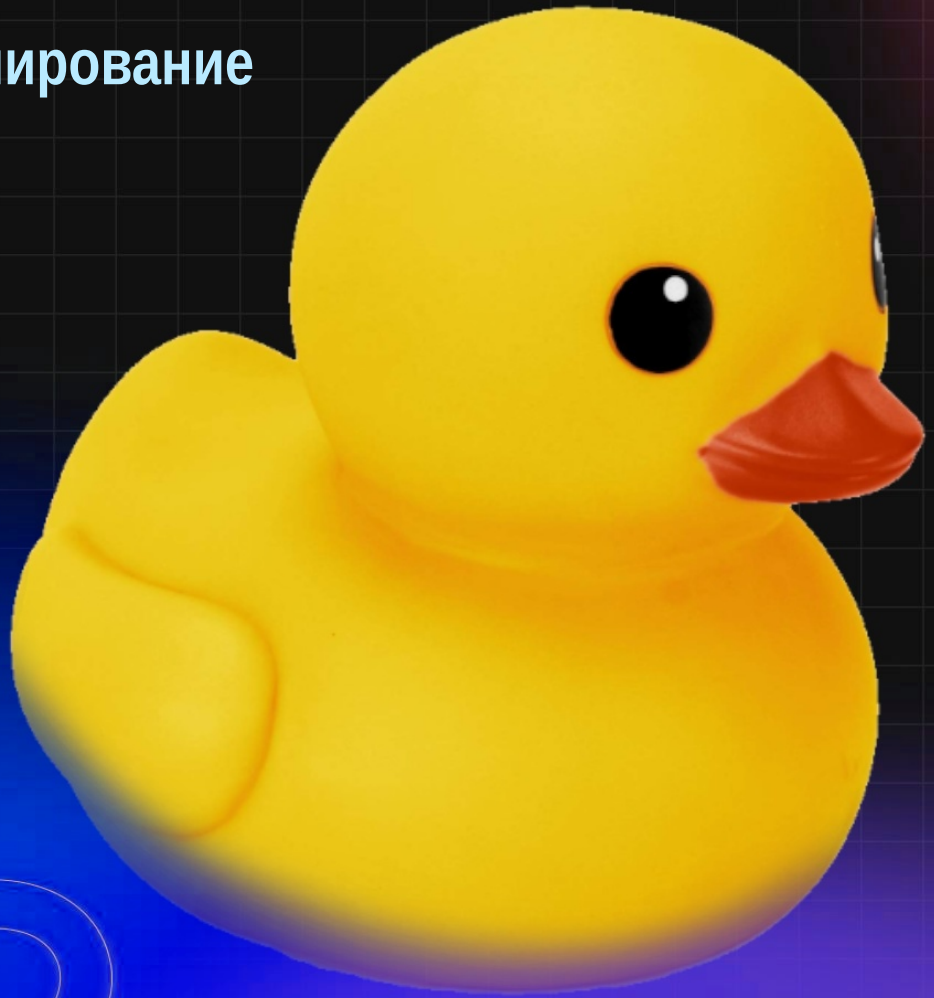
Использование аннотаций

ИТМО

- Проверка кода @Nullable, @Deprecated
- Методы для тестирования @Test, @BeforeAll, @AfterAll
- Хранение данных @Table, @Column, @Id
- Внедрение зависимостей @Autowired
- Графические библиотеки

Java

Программирование
2 семестр
2025



ИТМО

Полезные инструменты

Библиотеки и фреймворки

- Apache Commons
- Google Guava
- Lombok
- Jackson
- Spring
- ...

Apache Commons

- Независимые модули общего назначения
  - Apache Commons Lang 3
    - ObjectUtils, ArrayUtils, StringUtils, NumberUtils, SystemUtils
  - Apache Commons Collections
  - Apache Commons IO
    - IOUtils, FileUtils
  - Math, CSV, Codec, Compress
- commons.apache.org

Google Guava

ИТМО

- Единая библиотека - улучшение Java API
  - Коллекции
    - ImmutableList, ImmutableSet, ...
    - MultiSet, MultiMap, Table, BiMap, ...
  - Кэширование
  - Базовые утилиты
    - Splitter, Primitives, Preconditions, ...

Lombok

ИТМО

- Процессор аннотаций для генерации шаблонного кода
  - Плагин для IDE
  - @Getter, @Setter
  - @ToString, @EqualsAndHashCode
  - @Data
  - @Builder

Java

Использование ИИ

- Генерация и автодополнение кода
- Поиск ошибок и статический анализ
- Рефакторинг и оптимизация
- Генерация документации
- Объяснение кода
- Конвертация кода

Инструменты ИИ

- GitHub Copilot
- Intellij AI Assistant
- LLM общего назначения (ChatGPT, Gemini, Deepseek
- Специальные LLM (Codex, AlphaCode)

Возможные проблемы ИИ

- Галлюцинации
- Уникальная или сложная логика
- Безопасность кода
- Конфиденциальность
- Зависимость от ИИ