DIGITAL HABITS
EDUCATION

Основы работы Java программ

>_ java

Agenda

- JVM / JRE / JDK
- Classloaders / classpath
- Сборка Java приложений
- Классы / наследование / полиморфизм
- Инициализация полей и блоков класса
- Mutable vs immutable

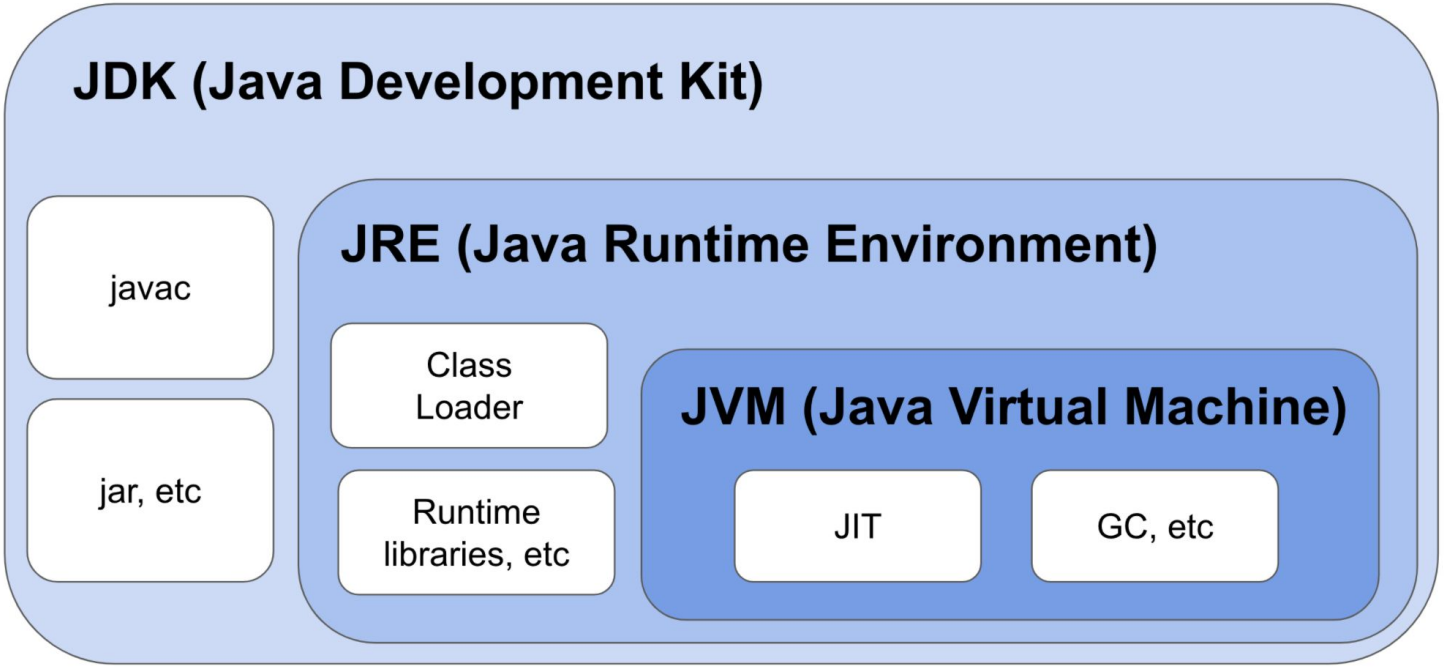
```
>_  java
```

JVM / JRE / JDK

- **JVM / JRE / JDK**
- Classloaders / classpath
- Сборка Java приложений
- Классы / наследование / полиморфизм
- Инициализация полей и блоков класса
- Mutable vs immutable

```
>_  java
```

JVM / JRE / JDK

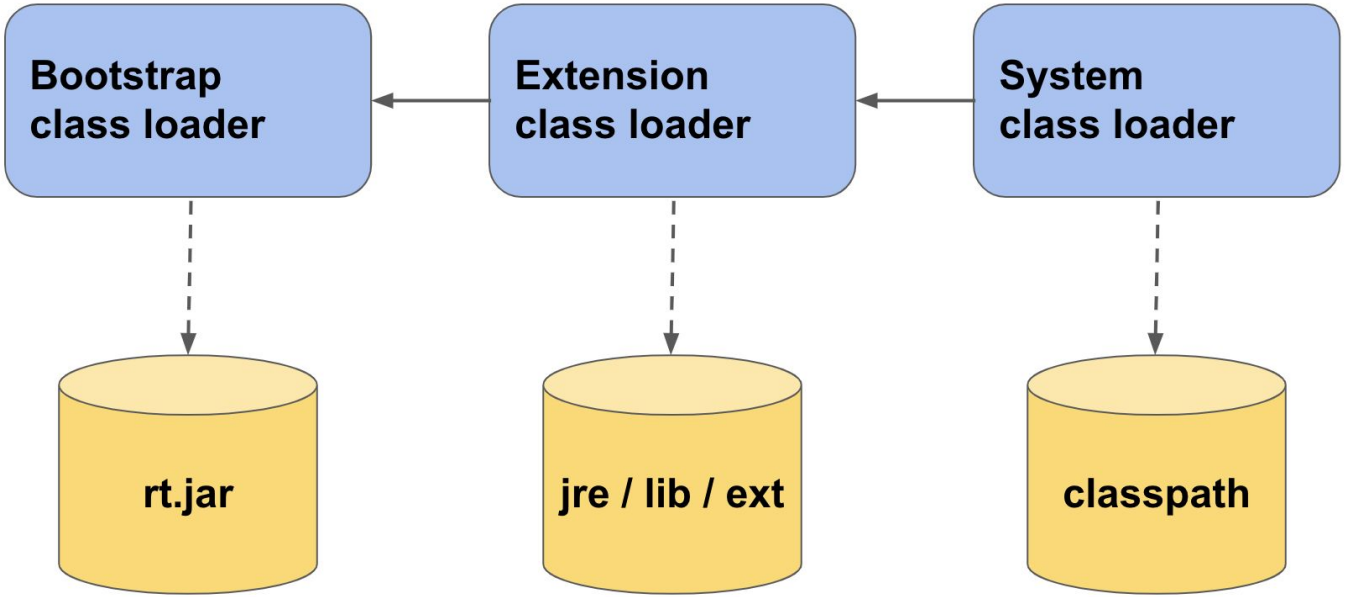

JDK (Java Development Kit)
javac
jar, etc
JRE (Java Runtime Environment)
Class Loader
Runtime libraries, etc
JVM (Java Virtual Machine)
JIT
GC, etc

Classloaders / classpath

- JVM / JRE / JDK
- **Classloaders / classpath**
- Сборка Java приложений
- Классы / наследование / полиморфизм
- Инициализация полей и блоков класса
- Mutable vs immutable

>_ java

Загрузка классов



Classpath

CLASSPATH указывает приложениям где искать пользовательские классы.

Устанавливать CLASSPATH можно в следующих местах:

1. В глобальных переменных операционной системы.

2. Дополнительным аргументом в утилите cli java.
   java -cp HelloWorld.jar Hello, -cp сокращенно от -classpath

3. В манифесте jar-архива.

Сборка и запуск приложения

Запуск: ***java <Class>.java***

Компиляция: ***javac <Class>.java*** => <*Class*>.class

Сборка jar: ***jar cfm <JarName>.jar <manifest> <path>/*.class*** => <JarName>.jar

Посмотреть содержимое jar: ***jar tf <JarName>.jar***

Запуск: ***java -cp <JarName>.jar <MainClass>***

Классы / наследование / полиморфизм

- JVM / JRE / JDK
- Classloaders / classpath
- Сборка Java приложений
- **Классы / наследование / полиморфизм**
- Инициализация полей и блоков класса
- Mutable vs immutable

>_ java

Классы и объекты

Что такое класс?

Классы и объекты

Что такое объект?

Наследование

- В java нет множественного наследования.
- В иерархии классов конструкторы вызываются в порядке наследования, начиная с суперкласса и заканчивая подклассом.
- Производный класс имеет доступ ко всем методам и полям базового класса кроме тех, которые определены с модификатором **private**.

Наследование

- Производный класс может определять свои методы, а может переопределять методы, которые унаследованы от базового класса.

- Запретить наследование можно с помощью модификатора **final**.

Полиморфизм

Что такое полиморфизм?

Полиморфизм

Полиморфизм - возможность применения одноименных методов с одинаковыми или различными наборами параметров в одном классе или в группе классов, связанных отношением наследования.

Как достичь:

- изменить поведение методов родительского класса ("переопределение методов")
- создавать "одноименные методы" в одном классе ("перегрузка методов")

Инициализация полей и блоков класса

- JVM / JRE / JDK
- Classloaders / classpath
- Сборка Java приложений
- Классы / наследование / полиморфизм
- **Инициализация полей и блоков класса**
- Mutable vs immutable

>_ java

static

Когда инициализируются static поля и static блоки класса?

static

Когда инициализируются static поля и static блоки класса?

- Статические поля и блоки инициализируются при загрузке класса с помощью classloader.

Порядок инициализации полей и блоков

В общем случае порядок вызова блоков и конструктора следующий:

1. Статические поля
2. Статический блок инициализации
3. Нестатические поля
4. Не статический блок инициализации
5. Конструктор

```
>_  java
```

Порядок инициализации полей и блоков

При наследовании инициализация происходит в следующем порядке:

1. Статические поля класса Parent;
2. Статический блок инициализации класса Parent;
3. Статические поля класса Child;
4. Статический блок инициализации класса Child;
5. Нестатические поля класса Parent;
6. Нестатический блок инициализации класса Parent;
7. Конструктор класса Parent;
8. Нестатические поля класса Child;
9. Нестатический блок инициализации класса Child;
10. Конструктор класса Child.

Mutable vs immutable

- JVM / JRE / JDK
- Classloaders / classpath
- Сборка Java приложений
- Классы
- Статические переменные, блоки
- Порядок инициализации полей и блоков
- **Mutable vs immutable**

>_ java

Mutable vs immutable

Mutable object (изменяемый) - можно изменить состояние и поля после создания объекта.

Примеры: StringBuilder, java.util.Date и т.д.

Mutable vs immutable

Immutable object (неизменяемый) - нельзя ничего изменить после создания объекта.

- Подходит для целей кеширования, потому что вам не нужно беспокоиться об изменении значений.
- Является потокобезопасным.

Примеры: все классы-обертки над примитивными типами (String, Integer, Byte и т.д.) и др.

Mutable vs immutable

Как сделать класс immutable?

Mutable vs immutable

Как сделать класс immutable?

- Объявляем класс как **final**;
- Все поля класса **private final**;
- Не используем сеттеры, используем **только конструктор**;
- Для полей, которые НЕ являются примитивом или immutable классом, **в геттере необходимо возвращать копию объекта**.

ДЗ

- Повторить теорию по пройденному материалу
- Перед следующей лекцией будет мини-тест