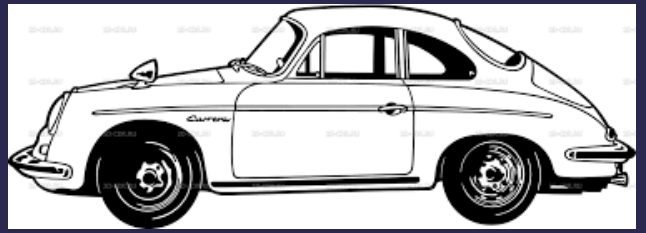
Курс Java-разработчик

ООП

Абстрактные классы

- Абстрактный метод – это не имеющий тела метод, который должен быть переопределён в классах наследниках.

В языке Java абстрактный метод объявляется с модификатором **abstract**, причём вместо тела метода ставится точка с запятой.



- Абстрактный класс – это класс, имеющий хотя бы один абстрактный метод.

Абстрактные классы

В программах на Java абстрактный класс объявляется с модификатором **abstract**. При этом экземпляры абстрактных классов создавать запрещено.

```java
package com.example;

public abstract class Animal {
    private String name;
    private Integer age;

    public Animal(String name, Integer age) {
        this.name = name;
        this.age = age;
    }

    public abstract void say();
}
```

Интерфейс

Интерфейс – это абстрактный класс, не имеющий полей, конструкторов и неабстрактных методов. В качестве базовых классов для интерфейса могут выступать только интерфейсы.
В Java для объявления интерфейса используется ключевое слово **interface**.

Интерфейс

```java
package com.example;

interface Movable {
    void move(float meters);
}

public abstract class Animal implements Movable {
    private String name;
    private Integer age;

    public Animal(String name, Integer age) {
        this.name = name;
        this.age = age;
    }

    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public Animal(String name, double age) {
        this(name, (int) age);
    }

    public abstract void say();
}
```

Перегрузка метода

- Перегрузка метода – это объявление для заданного класса двух или более методов, имеющих одинаковое имя, но различные параметры.

Решение о том, куда именно передаётся управление при вызове перегруженного метода X, принимается на этапе компиляции на основе сопоставления типов фактических параметров вызываемого метода и параметров, имеющих имя X.

Перегрузка метода

```java
package com.example;

public abstract class Animal {
    private String name;
    private Integer age;

    public Animal(String name, Integer age) {
        this.name = name;
        this.age = age;
    }

    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public Animal(String name, double age) {
        this(name, (int) age);
    }

    public abstract void say();
}
```

Домашнее задание (часть 2)

- Сделать класс Animal абстрактным.
- Все методы, которые общие для разных животных, сделать абстрактными.
- Методы, которые являются различными для всех животных, вынести в интерфейсы. (например, Moveable и т.д.)
- Класс Animal должен реализовать (имплементировать) эти интерфейсы.
- Добавить метод с перегрузкой (два метода с одинаковыми именами, но разными параметрами) для Animal.

Нештатные ситуации

- Нештатная ситуация – это ситуация, в которой выполнение некоторого фрагмента кода программы оказывается по тем или иным причинам невозможно.

Исключительные и ошибочные ситуации

- Исключительная ситуация – это нештатная ситуация, возникшая в силу внешних по отношению к программе причин.

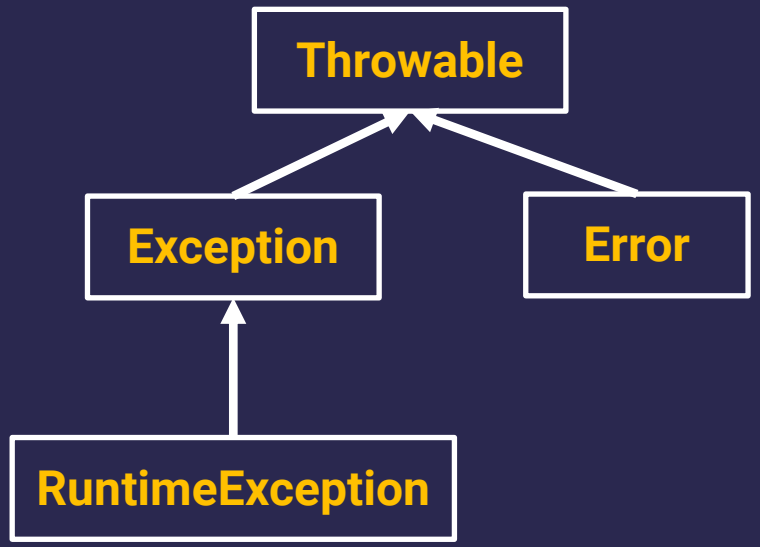
Примеры: не открылся файл, произошло незапланированное закрытие сетевого соединения.

- Ошибочная ситуация – это нештатная ситуация, возникшая из-за ошибки в коде программы.

Примеры: выход за границы массива; деление на ноль; переполнение стека.

Исключения

Исключение – это объект, описывающий нештатную ситуацию.



Иерархия исключений в Java

Throwable – базовый класс для всех классов исключений.
Error – базовый класс для классов исключений, описывающих «смертельные» для программы системные ошибочные ситуации
Exception – базовый класс для всех классов несистемных исключений.
RuntimeException – базовый класс для классов несистемных исключений, описывающих ошибочные ситуации.

Конструкторы класса **Throwable**

Основные конструкторы класса **Throwable**:

- Throwable(String message)
- Throwable(String message, Throwable cause)

Оба конструктора принимают в качестве параметра сообщение message, описывающее нештатную ситуацию.
Второй конструктор, кроме того, позволяет организовать так называемую «цепочку исключений», когда исключение (cause) – вложено в другое исключение, то есть подразумевается, что вложенное исключение является причиной объемлющего исключения.

Методы класса **Throwable**

- Throwable getCause ();
- String getMessage ();
- String toString ();
- void printStackTrace ();