Java Fundamentals

Основы объектно-ориентированного программирования

WebCamp

Объектно-ориентированное программирование

Наследование

Парадигмы ООП

ИНКАПСУЛЯЦИЯ

ПОЛИМОРФИЗМ

НАСЛЕДОВАНИЕ

АБСТРАКЦИЯ

Инкапсуляция

Парадигма ООП

Инкапсуляция — механизм языка, позволяющий ограничить доступ одних компонентов программы к другим.

В Java существуют следующие модификаторы доступа:

- private - члены класса доступны только внутри класса;
- default (package-private, модификатор по умолчанию) - члены класса видны внутри пакета;
- protected - члены класса доступны внутри пакета и в наследниках;
- public - члены класс доступны всем;

```
class A {
    private int field1;
    int field2;
    protected int field3;
    public int field4;
}
```

Наследование

Парадигма ООП

Наследование —механизм объектно-ориентированного программирования (наряду с инкапсуляцией, полиморфизмом и абстракцией), позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.

```
class A {
    public int field1;
    public void method() {
        /* … */
    }
}
```

```
class B extends A {
    public int field2;
}
```

```
static void main(String[] args) {
    B b = new B();
    b.field1 = 5;
    b.field2 = 8;
    b.method();
}
```

Полиморфизм

Парадигма ООП

Полиморфизм —возможность объектов с одинаковой спецификацией иметь различную реализацию.

Формы полиморфизма:

1. Ad-hoc полиморфизм
2. Классический (принудительный) полиморфизм:
    - использование переопределенных членов (@Override).
    - приведение типов.

Абстракция

Парадигма ООП

Абстракция в ООП – это придание объекту характеристик, которые отличают его от всех других объектов, четко определяя его концептуальные границы.

Абстрагирование в ООП – это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые. Соответственно, абстракция —это набор всех таких характеристик.

Объявление класса

Содержание класса

```
public class Car {

    public String model;
    private int maxSpeed;
    private int speed;

    public Car(String model, int maxSpeed, int speed) {
        this.model = model;
        this.maxSpeed = maxSpeed;
        this.speed = speed;
    }

    public int getMaxSpeed() {
        return maxSpeed;
    }
    public void setMaxSpeed(int maxSpeed) {
        this.maxSpeed = maxSpeed;
    }
}
```

Объявление класса с именем Car

Объявление полей класса Car

Объявление конструктора класса Car

Объявление методов класса Car

Члены класса

Поля и Методы

Поле класса или атрибут (class field) — переменная, связанная с классом или объектом. Все данные объекта хранятся в его полях. Доступ к полям осуществляется по их имени.

```
private int speed;
speed = 100;
```

Метод класса (class method) – процедуры и функции, связанные с классом. Определяют действия, которые можно выполнять над объектом такого типа, и которые сам объект может выполнять.

```
public int getSpeed(){
   return this.speed;
}
```

Конструкторы

Конструктор – специальный метод для создания экземпляра класса (инициализации объекта). Имя конструктора соответствует имени класса.

```
public Car(String model, int maxSpeed, int speed) {
      this.model = model;
      this.maxSpeed = maxSpeed;
      this.speed = speed;
  }
```

Если программист явно не задал конструктор в классе, JVM при компиляции создаст конструктор по умолчанию

```
public Car() {   }
```

Объект

Создание объекта

Чтобы создать объект используется ключевое слово new.

```
new Car();
```

При Этом JVM вызывает конструктор класса Car и собирает экземпляр класса Car.
Чтобы использовать объект в дальнейшем, создадим указатель на него

```
Car car = new Car();
```

Теперь через ссылку мы можем получить доступ к полям и методам объекта

```
car.model;
car.setMaxSpeed(100);
```

Значения полей характеризуют СОСТОЯНИЕ экземпляра класса (объекта)
Набор методов характеризует ПОВЕДЕНИЕ объекта.

Q&A