Полиморфизм

Полиморфизм в ООП

- ~~Инкапсуляция~~
- ~~Наследование~~
- Полиморфизм <- вы тут

Полиморфизм — это принцип объектно-ориентированного программирования, который позволяет объектам разных классов обрабатываться через общий интерфейс за счет наследования и переопределения (обладают “созвучными” методами от родительского класса, но поведение переопределенно).

Этот принцип позволяет одному и тому же методу выполнять различные функции в зависимости от объекта, к которому он применяется.

Пример

```java
package l25;

class Animal {
    void makeSound() {
        System.out.println("Some sound");
    }
}

class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Woof");
    }
}

class Cat extends Animal {
    @Override
    void makeSound() {
        System.out.println("Meow");
    }
}
```

Представьте, что у вас есть базовый класс Animal и производные классы Dog и Cat.

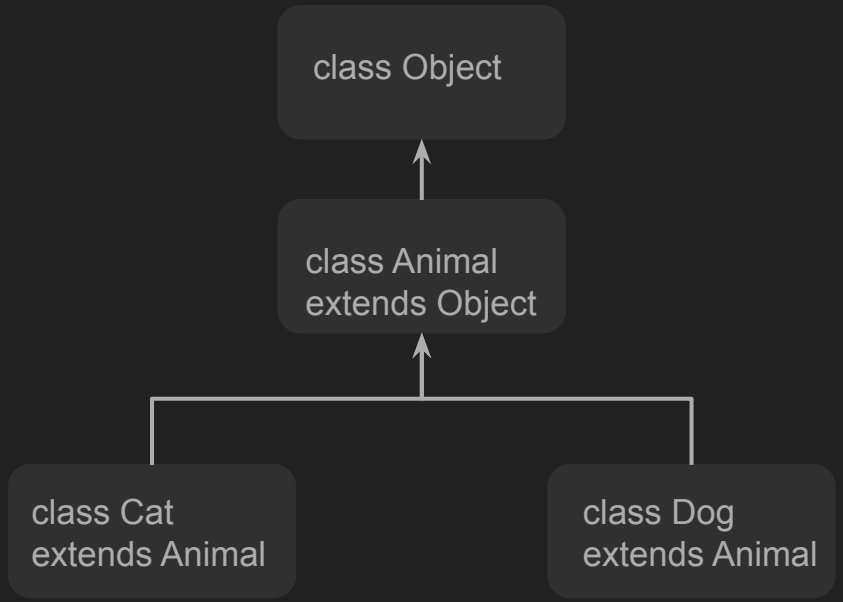
У всех животных есть метод makeSound, но звук, который издают собака и кошка, различается.

Контрольная точка

- Понятно ли что такое полиморфизм?
- Понятно ли основная его мысль?

Если все ясно, ставим плюсы, иначе - задаем вопросы.

Восходящее преобразование



Восходящее преобразование — это когда ссылка на объект подкласса присваивается переменной базового (родительского) класса.

Это безопасно, потому что подкласс всегда является более специфическим, чем базовый класс, и обладает всеми его свойствами и методами.

Подробнее читайте в [этой статье].

Пример

```
package l25;

class AnimalsExample {
   public static void main(String[] args) {
       Animal myDog = new Dog();
       Animal myCat = new Cat();
       Object myAnimal = new Animal();
       Object myCatObj = new Cat();
   }
}
```

В примере слева myDog и myCat являются объектами Animal, но фактически они указывают на объекты Dog и Cat соответственно. Это позволяет нам обращаться с разными объектами единообразно.

Контрольная точка

- Понятно ли что такое восходящее преобразование?
- Понятно ли как его использовать?

Если все ясно, ставим плюсы, иначе - задаем вопросы.

Переопределение методов

```java
package l25;

class Animal {
    void makeSound() {
        System.out.println("Some sound");
    }
}

class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Woof");
    }
}

class Cat extends Animal {
    @Override
    void makeSound() {
        System.out.println("Meow");
    }
}
```

Переопределение методов позволяет классу-наследнику предоставить специфическую реализацию метода, который уже определен в его базовом классе.

Это ключевая особенность, которая поддерживает полиморфизм, позволяя одному и тому же методу вести себя по-разному в зависимости от объекта, к которому он применяется.

Использование полиморфизма

```
class PolymorphismExample {
    public static void main(String[] args) {
        Animal[] animals = {new Dog(), new Cat()};

        for (Animal animal : animals) {
            // Выведет "Woof" для собаки
            // и "Meow" для кошки
            animal.makeSound();
        }
    }
}
```

Мы можем использовать полиморфизм для выполнения одного и того же действия разными способами с разными объектами, используя единственный интерфейс.

Попробуем вызывать метод makeSound на любом животном.

Контрольная точка

- Понятно ли как использовать полиморфизм?

Если все ясно, ставим плюсы, иначе - задаем вопросы.

Домашнее задание

доделать домашку 24

The end