Наследование

Возвращаемся к ООП

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

Наследование - это концепция, согласно которой одни классы, называемые родительскими, могут лежать в основе других — дочерних. При этом, дочерние классы перенимают свойства и поведение своего родителя.

Наследование наглядно

Что общего между двумя объектами реального мира?

Наследование наглядно

Поиск одинаковых свойств:

- имя
- возраст
- рост
- вес
- пол

Поиск одинакового поведения:

- сказать фразу
- применить навык
- присесть
- привстать

Наследование наглядно

```java
package l23.people;

public class Human {
    String name;  // имя

    // пол, если true - мужчина, иначе -
женщина
    boolean isMale;

    int age;  // возраст

    // ... остальные свойства

    public void sayPhrase() {
        System.out.println("Меня звать " +
name);
    }

    public void applySkill() {
        System.out.println("Я ничего не умею");
    }

    // ... остальное поведение (методы)
```

Поиск одинаковых свойств:

- имя
- возраст
- рост
- вес
- пол

Поиск одинакового поведения:

- сказать фразу
- применить навык
- присесть
- привстать

Наследование наглядно

```java
package l23.people;

public class BadBoy extends Human {
    public BadBoy(String name, int age) {
        // всегда true для BadBoy
        isMale = true;
        this.name = name;
        this.age = age;
    }

    public void applySkill() {
        takePhone();
    }

    public void takePhone() {
        System.out.println("Я отбираю ваш
телефон");
    }
}
```

```java
package l23.people;

public class BadGirl extends Human {
    public BadGirl(String name, int age) {
        // всегда false для BadGirl
        isMale = false;
        this.name = name;
        this.age = age;
    }

    public void applySkill() {
        System.out.println("Я открываю пиво глазом");
    }

}
```

Наследование наглядно

```
package l23.people;

public class Main {
    public static void main(String[] args) {
        Human[] people =  new Human[2];
        people[0] = new BadBoy("Вася", 29);
        people[1] = new BadGirl("Мурка", 27);

        for(Human human: people) {
            human.sayPhrase();
            human.applySkill();
        }
    }
}
```

Объекты дочерних классов можно в том числе использовать в качестве объектов родительских классов.

при наличии общего родителя мы можем работать с объектами обобщенно, как показанно в примере кода.

При этом через переменную с родительским типом доступны лишь те поля и методы, которые описаны в родительском методе.

Наследование: родительский конструктор

```java
package l23.people.v2;

public class Human {
    // ... старые поля

    public Human(String name, boolean isMale, int age) {
        this.name = name;
        this.isMale = isMale;
        this.age = age;
    }

    // ... старые методы
}
```

```java
package l23.people.v2;

public class BadGirl extends Human {
    public BadGirl(String name, int age) {
        // значение для isMale всегда false для BadGirl
        super(name, false, age);
    }

    // ... старые методы
}
```

Добавим конструктор в родительский класс.

Соответственно надо будет поменять тело констуктора в дочерних классах.

Для BadBoy все аналогично.

super() - вызов конструктора класса-родителя.

Про разницу между this и super: тут

Наследование: super

```java
package l23.people.v2;

public class BadGirl extends Human {
    // ... старый код

    public void applySkill() {
        super.applySkill();
        System.out.println("зато я красивая");
    }
}
```

Из метода дочернего класса можно вызывать методы описанные в родительском классе.

Для этого используется запись super.method, где method - название любого метода родительского класса.

Наследование: protected

МОДИФИКАТОР	КЛАСС	ПАКЕТ	ДЛЯ ВСЕХ
public	Да	Да	Да
отсутствует	Да	Да	Нет
protected	Да + в дочерних	Да	Нет
private	Да	Нет	Нет

- доступно из методов самого класса
- доступно в рамках всего пакета где лежит класс
- доступно в дочерних классах даже в другом пакете
- недоступно в других пакетах

Домашнее задание

доделать домашку 22

выделить общее поведение и свойства у дочерних классов (если уже выделены общие свойства и/или поведение, то добавьте в дочерние класс(ы) НЕ общие свойства и/или поведение), перенести их в родительский класс, при этом переопределить вызов одного метода в дочернем классе, а так же все поля должны быть protected.

Вариант 1: на основе 22 домашки основанной на 17 домашке (people, transport, animal)

Вариант 2: на основе 22 домашки основанной на Покемонах

The end