День 9

Темы:

Урок 25: Наследование
Урок 29: Полиморфизм
Урок 41: Абстрактные классы

Доп.материалы:
Ключевое слово `super`:
https://www.examclouds.com/ru/java/java-core-russian/keyword-super

Пример использования ключевого слова `super`:

Предположим, что у нас есть класс Компьютер (англ. `Computer`) с полями `manufacturer` и `productionYear` - название производителя и год производства компьютера. Также, у нас есть класс Ноутбук (англ. `Laptop`), который наследуется от класса `Computer` (потому что ноутбук является компьютером). У Ноутбука есть свое поле `batteryCapacity` - емкость аккумулятора. Для всех полей определены конструкторы.

```
class Computer {
    private String manufacturer;
    private int productionYear;

    public Computer(String manufacturer, int productionYear) {
        this.manufacturer = manufacturer;
        this.productionYear = productionYear;
    }
}

class Laptop extends Computer {
    private int batteryCapacity;

    public Laptop(int batteryCapacity) {
        this.batteryCapacity = batteryCapacity;
    }
}
```

Такой код выдаст ошибку. Почему?
Ноутбук наследуется от Компьютера. Следовательно, класс Ноутбук наследует все поля класса Компьютер (у ноутбука тоже есть название производителя и год производства).
Несмотря на то, что в классе `Laptop` мы не можем получить доступ к полям `Computer` (потому что они приватные), эти поля все равно инициализируются в процессе создания объекта класса `Laptop`.

Так как в классе `Computer` определен лишь один конструктор (который принимает на вход 2 аргумента), мы можем лишь одним способом создать объект класса `Computer` (передав на вход конструктору два значения - название и год).
Во время создания объекта класса `Laptop`, сначала инициализируются поля родительского класса (класса `Computer`), а затем уже инициализируется поле самого класса `Laptop`. Это означает, что при создании объекта класса `Laptop`, сначала должен вызваться конструктор `Computer`, а уже затем должен выполниться конструктор `Laptop`.
В коде выше мы никак не вызываем конструктор `Computer` при создании объекта `Laptop`, соответственно не инициализируются нужные нам поля Компьютера. Поэтому и возникает ошибка.
Чтобы исправить эту ошибку, необходимо использовать ключевое слово `super`. Это ключевое слово позволяет обращаться к классам, стоящим выше в иерархии наследования (к так называемым суперклассам).
Используя ключевое слово `super`, мы должны вызвать конструктор `Computer` внутри конструктора `Laptop` и передать в этот конструктор значения, относящиеся к “компьютерной” составляющей ноутбука.
Вот пример корректного кода:

```
class Computer {
    private String manufacturer;
    private int productionYear;

    public Computer(String manufacturer, int productionYear) {
        this.manufacturer = manufacturer;
        this.productionYear = productionYear;
    }
}

class Laptop extends Computer {
    private int batteryCapacity;

    public Laptop(String manufacturer, int productionYear, int batteryCapacity) {
        super(manufacturer, productionYear);
        this.batteryCapacity = batteryCapacity;
    }
}
```

Теперь, значения, передаваемые при создании в конструктор `Laptop`, передаются выше в конструктор `Computer`. Таким образом, при создании объекта класса `Laptop` будут вызваны два конструктора - конструктор Компьютера и конструктор Ноутбука. Соответственно, будут проинициализированы и поля, относящиеся к “Компьютерной” составляющей, и поле, относящееся только к Ноутбуковой части (емкость аккумулятора).

Пример создания нового объекта класса Laptop:

```
Laptop laptop = new Laptop("Apple", 2019, 4000);
```

Похожим образом, с помощью ключевого слова `super` можно вызывать методы родительских классов внутри методов классов-потомков.

Пример кода:

```
class A {
    public void testA() {
        System.out.println("Hello from class A");
    }
}

class B extends A {
    public void testB() {
        super.testA(); // вызываем метод testA() из родительского класса A
        System.out.println("Hello from class B");
    }
}
```

Задачи:

1. Создайте класс Человек (англ. `Human`). У человека должно быть поле “имя” (англ. `name`). На это поле в классе должен быть конструктор, `get` и `set` методы. Также, у Человека должен быть метод `printInfo()`, который выводит в консоль информацию о человеке в формате: `“Этот человек с именем ИМЯ”`.
Затем, создайте класс Студент (англ. `Student`), который наследуется от класса Человек. У студента есть дополнительное строковое поле - название его учебной группы. Для этого поля тоже необходимо создать геттер и сеттер. Конструктор в классе Студент должен принимать на вход два аргумента - имя и название учебной группы. Метод `printInfo()` в классе Студент должен быть переопределен таким образом, чтобы формат выводимого в консоль сообщения был таким:
`“Этот человек с именем ИМЯ”`
`“Этот студент с именем ИМЯ”`
(должно выводиться именно две строки - необходимо использовать ключевое слово `super`)
И наконец, в нашей программе должна быть еще одна сущность - Преподаватель (англ. `Teacher`). Преподаватель должен тоже наследоваться от класса Человек. При этом, у преподавателя есть дополнительное строковое поле - название предмета, который ведет этот преподаватель. Для этого поля необходимо создать `get` и `set` методы. Конструктор в классе Преподаватель должен принимать на вход два

аргумента - имя преподавателя и название преподаваемого предмета. Метод `printInfo()` в классе Преподаватель должен быть переопределен таким образом, чтобы формат выводимого в консоль сообщения был таким:
`"Этот человек с именем ИМЯ"`
`"Этот преподаватель с именем ИМЯ".`
(должно выводиться именно две строки - необходимо использовать ключевое слово `super`)

Создайте в методе `main()` класса `Task1` объект класса Студент и объект класса Преподаватель. Выведите в консоль название учебной группы у объекта-студента и название предмета у объекта-преподавателя. Затем, вызовите `printInfo()` на объектах и посмотрите на результат.

2. Создайте абстрактный класс Фигура (англ. Figure). Этот класс будет являться абстрактным представлением геометрической фигуры в нашей программе.

У всех фигур в нашей программе есть цвет, поэтому в классе Фигура есть строковое поле `color`. Создайте конструктор, геттер и сеттер для этого поля.

У класса Фигура определены два абстрактных метода:
```
public abstract double area();
public abstract double perimeter();
```

Реализация первого метода должна возвращать площадь фигуры, а реализация второго метода должна возвращать периметр фигуры.

Создайте 3 геометрические фигуры: Круг (англ. `Circle`), Прямоугольник (англ. `Rectangle`) и Треугольник (англ. `Triangle`). Каждая из фигур должна наследоваться от абстрактного класса Фигура (англ. `Figure`).

У класса Круг будет одно поле - радиус окружности.

У класса Прямоугольник будет два поля - ширина и высота.

У класса Треугольник будет три поля - длина каждой из сторон.

Для каждого из этих трех классов необходимо реализовать конструктор, который принимает на вход размерности фигуры и цвет фигуры.

В этих же классах, вам необходимо реализовать два метода (`area()` и `perimeter()`). Реализация этих методов будет разной для каждой из геометрических фигур. Формулы для вычисления площади и периметра легко находятся в интернете.

После того, как геометрические фигуры будут полностью реализованы, вам необходимо будет реализовать еще один класс - `TestFigures`. В этом классе, в

методе `main()` должны быть созданы и помещены в массив следующие геометрические фигуры:

```
Figure[] figures = {
        new Triangle(10, 10, 10, "Red"),
        new Triangle(10, 20, 30, "Green"),
        new Triangle(10, 20, 15, "Red"),
        new Rectangle(5, 10, "Red"),
        new Rectangle(40, 15, "Orange"),
        new Circle(4, "Red"),
        new Circle(10, "Red"),
        new Circle(5, "Blue")
};
```

В этом же классе реализуйте два метода:

```
public static double calculateRedPerimeter(Figure[] figures)
public static double calculateRedArea(Figure[] figures)
```

Первый метод, принимая на вход массив геометрических фигур, должен вернуть сумму периметров красных фигур. Второй метод, принимая на вход массив геометрических фигур, должен вернуть сумму площадей красных фигур.

Вызовите эти два метода на массиве `figures` и выведите результат в консоль.