
DataArt

DataArt

Java-интенсив

DataArt

Про домашнее задание

- Называете переменные по смыслу
- Лишние комментарии только усложняют чтение кода
- Стоит иногда думать об оптимальности кода
- Не пишите весь код Main классе
- Рекурсия - зло

DataArt

Наиболее красивый код

- Ирина Просина
- Анна Щербакова
- Денис Вдовенко

DataArt

Прислали ДЗ

- Светлана Зелинская
- Яна Крылова
- Андрей Болдырев
- Влад Цыбизов
- Носенко Влад
- Юлия Бессонова
- Максим Булычев
- Мишель Гилавоги
- Максим Дуров
- Алексей Железной
- Владимир Менжунов
- Михаил Дмитриев
- Алина Горбунова
- Артем Ткаченко
- Савщинко Дмитрий

DataArt

Chapter 2.1

Поговорим про ООП

Объектно-ориентированное программирование

Класс

Состояние (свойства)

```
public class Human {

    Integer age;
    String name;

    public void say() {
        System.out.println("My name is " + name + " and I am " + age + " years old");
    }

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

Поведение (метод)

```
public class Main {

    public static void main(String[] args) {

        Human alex = new Human(23, "Alex");
        Human james = new Human(42, "James");

        alex.say();
        james.say();

    }
}
```

Вызовы методов

```
Main
"C:\Program Files\Java\jdk1.8.0_25\bin\java"
My name is Alex and I am 23 years old
My name is James and I am 42 years old

Process finished with exit code 0
```

DataArt

Объектно-ориентированное программирование

- Полиморфизм
- Инкапсуляция
- Наследование

DataArt

ООП : наследование

- Возможность создавать классы-потомки
- Классы-потомки наследуют поведение и состояние предка
- В потомке можно добавлять новое поведение или состояние
- Можно модифицировать унаследованное поведение или состояние

ООП : наследование

```
public class Student extends Human {

    private Integer course;

    public void prepareToExam() {
        if (course <= 2) {
            System.out.println("I will learn lectures!");
        } else {
            System.out.println("No lectures! I will cheat!");
        }
    }

    public Student(Integer age, String name, Integer course) {
        super(age, name);
        this.course = course;
    }
}
```

Новое состояние и поведение

DataArt

ООП : наследование

```
public class Main {

    public static void main(String[] args) {

        Student newStudent=new Student( age: 18,  name: "Alex",  course: 1);
        Student oldStudent=new Student( age: 22,  name: "Max",  course: 4);

        newStudent.say();
        newStudent.prepareToExam();

        oldStudent.say();
        oldStudent.prepareToExam();

    }
}
```

```
Main
"C:\Program Files\Java\jdk1.7.0_80\bin\java" ...
My name is Alex and I am 18 years old!
I will learn lectures!
My name is Max and I am 22 years old!
No lectures! I will cheat!
```

ООП : наследование

```java
public class Batman extends Human {

    @Override
    public void say() {
        System.out.println("I am Batman");
    }

    public Batman() { super( age: 40,  name: "Bruce Wayne"); }

}
```

Переопределение поведения

```java
public class Main {

    public static void main(String[] args) {

        Batman batman = new Batman();
        batman.say();

    }
}
```

```
Main
"C:\Program Files\Java\jdk1.7.0_80\bin\java" ...
I am Batman
```

DataArt

ООП : инкапсуляция

- Упаковка данных и функций в единый компонент
- Скрывает детали реализации от пользователя класса
- Делает систему более гибкой

ООП : инкапсуляция

```
public class Batman extends Human {

    private Integer batarangCount = 3;

    public Batman() {
        super( age: 40,  name: "Bruce Wayne");
    }

    public void throwBatarang() {
        if (batarangCount > 0) {
            System.out.println("Batarang has been thrown");
            batarangCount--;
        } else {
            System.out.println("no more batarang's");
        }
    }
}
```

Поле недоступно извне

DataArt

ООП : инкапсуляция

```
public class Main {

    public static void main(String[] args) {

        Batman batman = new Batman();
        batman.say();

        for (int i = 0; i < 4; i++) {
            batman.throwBatarang();
        }

        batman.
    }
}
```

```
m throwBatarang()                                   void
m say()                                             void
m equals(Object obj)                                boolean
m hashCode()                                        int
m toString()                                        String
m getClass()                          Class<? extends Batman>
m notify()                                          void
m notifyAll()                                       void
m wait()                                            void
m wait(long timeout)                                void
```

Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >>

```
Main
"C:\Program Files\Java\jdk1.7.0_80\
I am Batman
Batarang has been thrown
Batarang has been thrown
Batarang has been thrown
no more batarang's
```

Поле недоступно извне

DataArt

Модификаторы доступа

- Public
- Protected
- Private
- Package Private

DataArt

ООП : полиморфизм

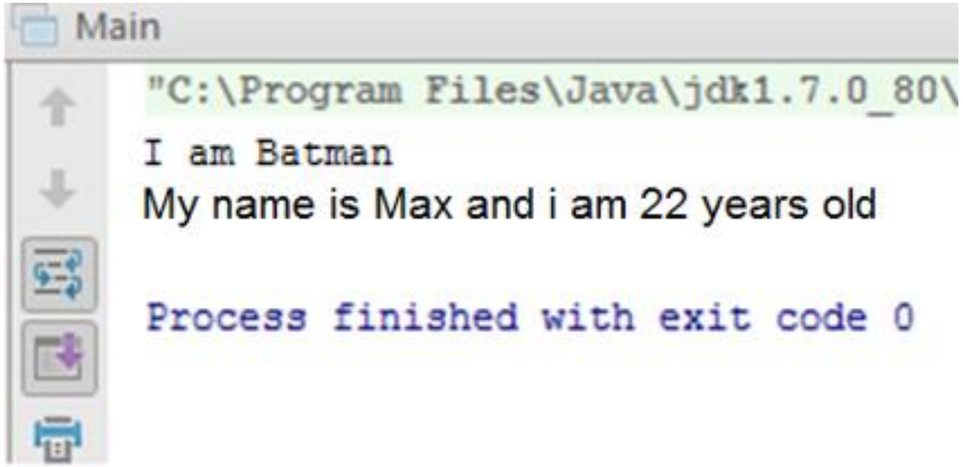
- С точки зрения ООП – наличие у объектов-потомков методов, одинаковых по сигнатуре, но различных по реализации
- Позволяют использовать похожие алгоритмы с различной реализацией в одном конкесте.

Объектно-ориентированное программирование

DataArt

```
public class Main {

    public static void main(String[] args) {

        Human humanStudent = new Student( age: 22,  name: "Max",  course: 4);
        Human humanBatman = new Batman();

        Human[] humansArr = new Human[2];
        humansArr[0] = humanBatman;
        humansArr[1] = humanStudent;

        for (Human human : humansArr) {
            human.say();
        }

    }
}
```

Конкретный тип переменной human не известен на каждом этапе цикла.



Зачем вообще нужно ООП?

- Код разбивается на большее число модулей
- Модули менее зависимы
- Связи между модулями не зависят от реализации

У меня вопрос!
~~Оказавшись перед Гослингом, что вы ему скажете~~
Я вот создал класс,
ни от кого его не наследовал, ничего не переопределял или добавлял, а у него все равно есть какие то методы.
toString, equals итд... Откуда?

DataArt

DataArt

```
public class NewClass {
}

public class Main {

    public static void main(String[] args) {
        NewClass newClass = new NewClass();
        newClass.
    }
}
```

```
equals(Object obj)                                  boolean
hashCode()                                          int
toString()                                          String
getClass()                                          Class<? extends NewClass>
notify()                                            void
notifyAll()                                         void
wait()                                              void
wait(long timeout)                                  void
wait(long timeout, int nanos)                       void
field                                               myField = expr
```

Ctrl+Down and Ctrl+Up will move caret down and up in the editor >>

Мир Java Core.
Класс Object

DataArt

- Вершина иерархии всех объектов
- На объекте основывается ООП в java
- Все методы, которые есть у объекта, будут у любого созданного класса.

Мир Java Core.
Класс Object

DataArt

- toString - превращает объект в строку
- equals – сравнивает объекты
- hashCode – возвращает хеш объекта
- Ну и еще многопоточные методы, getClass, но об этом не сегодня

Класс Object, Методы

DataArt

- toString - превращает объект в строку

```
public class User {

    private int id;
    private String name;
    private String password;

    public User(int id, String name, String password) {
        this.id = id;
        this.name = name;
        this.password = password;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }
}
```

Класс Object, Методы

DataArt

- toString - превращает объект в строку

```
public class Main {

    public static void main(String[] args) {
        User user = new User(42, "Daniel", "SecretPassword");
        System.out.println(user.toString());
    }


}
```

```
"C:\Program Files\Java\jdk1.8.0_25\bin\java" ...
User{id=42, name='Daniel'}

Process finished with exit code 0
```

Класс Object, как определить equals

DataArt

- Сравнивает объекты
- Если a.equals(b), то b.equals(a)
- a.equals(a)
- Если a.equals(b), и b.equals(c), то a.equals(c)
- Если объекты не изменялись, equals для них всегда возвращает один и тот же результат
- a.equals(null) возвращает false

Класс Object, Методы

DataArt

- hashCode– вычисляет ‘хеш код объекта’
- Пример очень простой хеш функции – сложить все ‘буквы’ в строке, и взять остаток от деления на 10
- Для равных объектов хеш всегда равный
- Для разных объектов, хеш почти всегда разный, за исключением коллизий
- Часто используется для ‘ускорения’ операций

DataArt

Класс Object, Методы

- строка1 = “abc”
- строка2 = “abd”
- строка3 = “cba”
- Какие будет считаться для них hashcode?

Домашнее задание

DataArt

- Создать иерархию из 1 предка и 3х наследников этого предка (пример, Музыкальные инструменты, Бытовая Техника. Животные)
- В одном наследнике переопределить equals и hashcode. Продемонстрировать работу этих методов
- Во всех наследниках переопределить toString. Продемонстрировать полиморфизм.
- У предка должен быть хотя бы 1 публичный метод, продемонстрировать
- У одного из наследников должен быть приватный метод. Использовать этот метод внутри публичного метода (можно создать еще один публичный метод, можно переопределить существующий)

DataArt

Вопросы и ответы


DataArt