
# Java

# Класс Object

Лекция #16

Пустовалова О.Г.
доцент. каф. мат.мод.
ИММиКН ЮФУ

# Содержание

# Класс Object

# Класс Object

В Java определен один специальный класс, называемый Object.

Все остальные классы являются подклассами, производными от этого класса, даже если в объявлении это явно не указано.

В классе Object определен ряд методов, которые доступны всем классам языка Java.

# Некоторые методы класса Object

- ✓ public String **toString**() - возвращает символьную строку, описывающую объект.
- ✓ public int **hashCode**() - возвращает хеш-код, связанный с вызывающим объектом.
- ✓ public boolean **equals**(Object obj) - определяет, равен ли один объект другому.
- ✓ protected Object **clone**() - создает новый объект, не отличающийся от клонируемого.

# Метод toString класса Object

# Методы класса Object. toString

Метод **toString** служит для получения представления данного объекта в виде строки.

```
public class Program {

    public static void main(String[] args) {

        Person tom = new Person("Tom");
        System.out.println(tom.toString()); // Person@3feba861
    }
}
class Person {

    private String name;

    public Person(String name){

        this.name=name;
    }
}
```

При попытке вывести строковое представления какого-нибудь объекта, как правило, будет выводиться полное имя класса.

# Методы класса Object. toString

```
public class Program{

    public static void main(String[] args) {

        Person tom = new Person("Tom");
        System.out.println(tom.toString()); // Person Tom
    }
}
class Person {

    private String name;

    public Person(String name){

        this.name=name;
    }

     @Override
     public String toString(){

         return "Person " + name;
    }
}
```

При создании нового класса принято переопределение toString() таким образом, чтобы возвращающая строка содержала в себе имя класса, имена и значения всех переменных.

# Метод hashCode класса Object

# Методы класса Object. hashCode

Метод **hashCode** позволяет задать некоторое числовое значение, которое будет соответствовать данному объекту или его хэш-код.

По данному числу, например, можно сравнивать объекты.

```
Person tom = new Person("Tom");

System.out.println(tom.toString()); // Person Tom

System.out.println(tom.hashCode()); // 189568618
```

# Методы класса Object. hashCode

```
class Person {

    private String name;

    public Person(String name){

        this.name=name;
    }


     @Override
     public int hashCode(){

         return 10 * name.hashCode() ;
     }
}
```

Можно задать свой алгоритм определения хэш-кода объекта.

# метод getClass класса Object

# Методы класса Object. getClass

Метод **getClass** позволяет получить тип данного объекта.

```
Person tom = new Person("Tom");

System.out.println(tom.toString()); // Person Tom

System.out.println(tom.getClass()); // class Person
```

# Метод equals класса Object

# Методы класса Object. Метод equals

Метод **equals** сравнивает два объекта на равенство

```
Person tom = new Person("Tom");

Person bob = new Person("Bob");


System.out.println(tom.equals(bob)); // false


System.out.println(tom.equals(tom)); // true
```

# Методы класса Object. Метод equals

```
public class Program{

    public static void main(String[] args) {

        Person tom = new Person("Tom");

        Person tom2 = tom;

System.out.println(tom.equals(tom2)); // true

System.out.println(tom.hashCode()-tom2.hashCode()); // 0

    }
}
class Person {

    private String name;

    public Person(String name){

        this.name=name;
    }}
```

Ссылка на один и тот же объект

.

# Методы класса Object. Метод equals

Оператор **instanceof** позволяет выяснить, является ли переданный в качестве параметра объект объектом определенного класса

```
Person tom = new Person("Tom");

System.out.println(tom instanceof Person); // true
```

# Методы класса Object. Метод equals

```
@Override
public boolean equals(Object obj){

    if (!(obj instanceof Person)) return false;

    Person p = (Person)obj;
    return this.name.equals(p.name);
}
```

Если объект не класса Person, то сравнение не производим.

```
Person tom = new Person("Tom");

Person tom2 = new Person("Tom");
System.out.println(tom.equals(tom2)); // true
```

Переопределили метод equals так что, если имена равны, то объекты равны.

# Метод clone класса Object

# Что такое клонирование

Иногда необходимо на основе существующего объекта создать второй такой же - то есть создать его клон. Это процесс в Java называется клонированием.

Для клонирования объекта в Java можно воспользоваться тремя способами:

1. Переопределение метода **clone**() и реализация интерфейса Cloneable().
2. Использование **конструктора копирования**.
3. Использовать для клонирования механизм **сериализации**.

# Сериализация

**Сериализация** - это процесс сохранения состояния объекта в последовательность байт.

**Десериализация** - это процесс восстановления объекта, из этих байт.

# Конструктор копирования

```
class Person {

    private String name;

    public Person(String name){

        this.name=name;
    }

    public String getName() {
        return name;
    }


     public Person(Person otherPerson){

         this(otherPerson.getName());
     }

}
```

Конструктор, принимающий на вход объект того же класса, который необходимо клонировать.

# Конструктор копирования

```
public class Program{

    public static void main(String[] args) {

        Person tom = new Person("Tom");
        Person tom2 = new Person(tom);

        System.out.println(tom.hashCode()==tom2.hashCode()); // false
    }
}
```

# Методы класса Object. Метод clone

```
class Person implements Cloneable{

    private String name;

    public Person(String name){

        this.name=name;
    }

    public String getName() {
        return name;
    }

    public Person clone()  throws CloneNotSupportedException {


        return (Person) super.clone();
    }
}
```

Класс *Object* определяет метод *clone(),* который создает копию объекта.

Если нужно, чтобы экземпляр вашего класса можно было клонировать, необходимо переопределить этот метод и реализовать интерфейс *Cloneable.*

*Object.clone()* выбрасывает исключение *CloneNotSupportedException* при попытке клонировать объект не реализующий интерфейс *Cloneable*.

# Методы класса Object. Метод clone

```
public class Program{

    public static void main(String[] args) throws CloneNotSupportedException
{

        Person tom = new Person("Tom");
        Person tom2 = tom.clone();

        System.out.println(tom.equals(tom2)); // false

        System.out.println(tom.hashCode()==tom2.hashCode()); // false
    }
}
```

# Спасибо за внимание!