Class `Object`

Class `Object`

Constructor

Constructor

```
/**
* Constructs a new object.
*/
@HotSpotIntrinsicCandidate
public Object() {}
```

Methods

getClass(): Class<?>

```
public class Test {
    public static void main(String[] args) {
        Object obj = new String("Hell to world!");
        Class c = obj.getClass();
        System.out.println("Class of Object obj is: " + c.getNa
        // Class of Object obj is: java.lang.String
    }
}
```

hashCode(): int

```
public class Program {
    public static void main(String[] args) {
        Person tom = new Person("Tom");
        System.out.println(tom.hashCode());
    }
}

class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    @Override
    public int hashCode() {
        return 19 * name.hashCode() + 7;
    }
}
```

toString(): String

```
public String toString() {
    return getClass().getName() + "@" + Integer.toHexString(has
}
```

toString(): String

```
class Complex {
    private double re;
    private double im;

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }

    @Override
    public String toString() {
        return String.format(re + " + i" + im);
    }
}
```

toString(): String

```
public class DEmo {
    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);
        System.out.println(c1);
    }
}
```

equals(Object): boolean

```
class Complex {
    private double re;
    private double im;

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }

    @Override
    public boolean equals(Object o) {
        if (o == this) {
            return true;
        } else if (!(o instanceof Complex)) {
            return false;
        } else {
            Complex c = (Complex) o;
            return (Double.compare(re, c.re) == 0) && (Double
        }
```

equals(Object): boolean

```
public class Demo {
    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);
        Complex c2 = new Complex(10, 15);
        if (c1.equals(c2)) {
            System.out.println("Equal ");
        } else {
            System.out.println("Not Equal ");
        }
    }
}
```

finalize(): void

```
public class Test {
    public static void main(String[] args) {
        Test t = new Test();
        System.out.println(t.hashCode());
        t = null;
        System.gc();
        System.out.println("end");
    }

    @Override //@Deprecated(since="9")
    protected void finalize() {
        System.out.println("finalize method called");
    }
}
```

Multithreading methods

- `notify(): void`
- `notifyAll(): void`
- `wait(): void`
- `wait(long): void`
- `wait(long, int): void`

How clone objects?

How clone objects?

- `clone(): Object`
- Interface `Cloneable` (**интерфейс-маркер/Marker Interface**)

```
public interface Cloneable {
}
```

clone(): Object

```
public class Person implements Cloneable {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Person{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }

    @Override
    protected Person clone() throws CloneNotSupportedExceptio
```

clone(): Object

```
public class Demo {
    public static void main(String[] args) throws CloneNotSuppo
        Person lucas = new Person("Lucas", 23);
        System.out.println(lucas);
        Person leo = lucas.clone();
        System.out.println(leo);
        System.out.println(lucas.equals(leo));
    }
}
```