ПРИМИТИВНЫЕ ТИПЫ JAVA

Autoboxing (автоупаковка)

ПРИМИТИВНЫЕ ТИПЫ JAVA

Boxing & Unboxing

- Boxing – автоматическое преобразование примитивного типа в соответствующий объект класса-обертки:

```
Integer a = 5;
```

- UnBoxing – обратное преобразование

```
Integer a = 5;
int b = a;
```

ВАЖНЫЕ ЗАМЕЧАНИЯ

```
public static void test1() {
	int a = 1;
	int b = 1;

	System.out.println("---------------");
	System.out.println(a > b);
	System.out.println(a < b);
	System.out.println(a == b);
}
//	false false true
```

ВАЖНЫЕ ЗАМЕЧАНИЯ

```
public static void test1() {
	Integer a = 1;
	Integer b = 1;

	System.out.println("---------------");
	System.out.println(a > b);
	System.out.println(a < b);
	System.out.println(a == b);
}
//	false false true
```

ВАЖНЫЕ ЗАМЕЧАНИЯ

```
public static void test1() {
        Integer a = 1000;
        Integer b = 1000;

        System.out.println("---------------");
        System.out.println(a > b);
        System.out.println(a < b);
        System.out.println(a == b);
}
//      false false false
```

ВАЖНЫЕ ЗАМЕЧАНИЯ

```
public static void test1() {
        Integer a = new Integer(1);
        Integer b = new Integer(1);

        System.out.println("---------------");
        System.out.println(a > b);
        System.out.println(a < b);
        System.out.println(a == b);
}
//      false false false
```

ПРОВЕРКА НА РАВЕНСТВО

В Java есть два способа сравнить объекты на равенство, == и метод equals.

- == используется для примитивных типов.
- Для объектов == это исключительно сравнение ссылок!

- Для остального надо использовать метод equals.
  Кроме того, метод hashCode служит (теоретически) для той же цели. Хорошим тоном считается переопределять его, если вы переопределили equals.
- Золотое правило сравнения:
  Если после инициализации неких объектов a и b выражение a.equals(b) вернёт true, то a.hashCode() должен быть равен b.hashCode().

КАК НЕ НУЖНО ПИСАТЬ

НЕЛЬЗЯ

```
long sum = 0;
for (Integer j = 0; j < 1000; j++) {
     for (Integer i = 0; i < 1000; i++) {
          sum += j*1000+i;
     }
}
```

Работает в 10! раз дольше, чем с int

Передача аргументов в методы

ПРИМИТИВНЫЕ ТИПЫ JAVA

ПЕРЕДАЧА АРГУМЕНТОВ

- Передача по ссылке подразумевает передачу ссылки на объект. В этом случае реализация метода потенциально может модифицировать переданный объект (например, вызвав метод, изменяющий состояние объекта).
- В случае передачи по значению параметр копируется. Изменение параметра не будет заметно на вызывающей стороне.
- В Java объекты всегда передаются по ссылке, а примитивы - по значению

ПРИМЕР

```
class Links1 {
	public static void foo(int x) {
		x = 3;
	}

	public static void main(String[] args) {
		int x = 1;
		foo(x);
		System.out.println(x);
	}
}
// 1
```

ПРИМЕР

```
class Links2 {
	public static void foo(Integer x) {
		x = 3;
	}

	public static void main(String[] args) {
		Integer x = 1;
		foo(x);
		System.out.println(x);
	}
}
// 1
```

ПРИМЕР

```
class Links3 {
	public static void foo(String x) {
		x = “3”;
	}

	public static void main(String[] args) {
		String x = “1”;
		foo(x);
		System.out.println(x);
	}
}
// 1
```

ПРИМЕР

```
class Links4 {
	public static void foo(String x) {
		x = new String(“3”);
	}

	public static void main(String[] args) {
		String x = “1”;
		foo(x);
		System.out.println(x);
	}
}
// 1
```

ПРИМЕР

```
class Links5 {
	public static void foo(Point p) {
		p.x = 3;
	}

	public static void main(String[] args) {
		Point p = new Point(0, 0);
		foo(p);
		System.out.println(p.x);
	}
}
// 3
```

Java

НЕ примитивные типы

String

НЕ ПРИМИТИВНЫЕ ТИПЫ JAVA

СТРОКИ

- Последовательность символов произвольной длины
- Класс java.lang.String
- Не то же, что массив символов
- Никаких нулевых символов в конце, длина хранится отдельно
- Строки не изменяемы!

- String hello = " Hello";
- String specialChars = "\r\n\t\"\\ \u0101\u2134\u03ff ";
- char[] charArray = {'a', 'b', 'c'};
- String string = new String (charArray);

ДОСТУП К СОДЕРЖИМОМУ

- int length()
- char charAt(int index)
- char[] toCharArray()

- String substring(int beginIndex)
- String substring(int beginIndex, int endIndex)

СРАВНЕНИЕ СТРОК

- Оператор == сравнивает ссылки, а не содержимое строки

- boolean equals(Object anObject)
- boolean equalsIgnoreCase(String anotherString)

- int compareTo(String anotherString)
- int compareToIgnoreCase(String anotherString)

ОПЕРАЦИИ

- boolean startsWith(String prefix)
- boolean endsWith(String suffix)
- int indexOf(String str)
- int lastIndexOf(String str)

- String trim()
- String replace(char oldChar, char newChar)
- String toLowerCase()
- String toUpperCase()

КОНКАТЕНАЦИЯ СТРОК

- String concat(String str)
- Оператор + (работает, как StringBuilder)

```
String helloWorld = " Hello " + " World!";
```

- java.lang.StringBuilder

```
StringBuilder buf = new StringBuilder();
buf.append ("Hello");
buf.append ("World");
buf.append ('!');
String result = buf.toString()
```

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

- Регулярные выражения поддерживаются в стандартной библиотеке Java

- boolean matches(String regex)
- String[] split(String regex)
- String replaceAll(String regex, String replacement)
- String replaceFirst(String regex, String replacement)

ПРИМЕРЫ

```
String str = "a, b, c,d, e";
String[] items = str.split (", *");
// items -> {"a", "b", "c", "d", "e"}

String str = "(aa)(bb)(cccc)";
String regex = "\\(([^)]*)\\)";
String result = str.replaceAll (regex , "$1");
// result -> "aabbcccc"
```