Java
Объектные оболочки и автоупаковка

Лекция #18

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

Содержание

Классы-оболочки

Классы-оболочки

Классы-оболочки Java являются Объектным представлением восьми примитивных типов в Java.

Примитивный тип	Класс-обертка	Аргументы
byte	Byte	byte или String
short	Short	short или String
int	Integer	int или String
long	Long	long или String
float	Float	float, double или String
double	Double	double или String
char	Character	char
boolean	Boolean	boolean или String

Классы-оболочки

У примитивных типов есть **объекты-аналоги** - **"классы оболочки"**, или **"wrapper"**.

Класс называется "оболочкой" потому, что он, по сути, копирует то, что уже существует, **но добавляет новые возможности для работы с привычными типами**.

Но поскольку это именно класс, он может создавать свои экземпляры.
Они будут хранить внутри нужные значения примитивов, при этом будут являться настоящими объектами.

Классы-оболочки

Объекты классов оберток создаются так же, как и любые другие:

```
Integer i = new Integer(682);

Double d = new Double(2.33);

Boolean b = new Boolean(false);
```

У классов-оболочек есть методы. У примитивных типов методов нет.

```
Integer i = new Integer(432);
String s = i.toString();
```

Классы-оболочки

```
System.out.println(Integer.MAX_VALUE);
System.out.println(Integer.MIN_VALUE);
```

```
2147483647
-2147483648
```

```
System.out.println(Float.MAX_VALUE);
System.out.println(Float.MIN_VALUE);
```

```
3.4028235E38
1.4E-45
```

```
System.out.println(Double.MAX_VALUE);
System.out.println(Double.MIN_VALUE);
```

```
1.7976931348623157E308
4.9E-324
```

Классы-оболочки

Примитивы и их аналоги, **классы оболочки**, существуют параллельно, потому что у каждого есть преимущества.

Обычный **int** занимает меньше места.

В свою очередь, с помощью класса-оболочки **Integer** можно выполнять специальные операции - например, перевести текст в число (с помощью метода **.parseInt()** для Integer-а).

```
Integer x = Integer.parseInt("1234");
int y = Integer.parseInt("567");
```

Зачем нужны классы-оболочки

Когда использовать примитивные типы и классы-оболочки:

- ✓ Используйте классы-обертки, когда работаете с **коллекциями**.
- ✓ Используйте примитивные типы для того, чтобы ваши программы были **максимально просты**.

Примитивные типы **не могут** быть **null**, а классы-оболочки — **могут**.

```
int x = null;

Integer y = null;
```

Автоупаковка
autoboxing

Автоупаковка

Автоупаковка (autoboxing) — это автоматическая конвертация из примитивных типов в соответствующий этому типу класс-обёртку, вставляемая компилятором Java, например из **float** во **Float** , из **int** в **Integer**.

```
Character ch = 'a';
Integer i1 = 220;
Double d1 = 300.0;
Boolean b1 = false;
```

Параметры передаются в методы по-разному:

- примитивы — по значению,
- а объекты — по ссылке.

Когда происходит автоупаковка

- При передаче примитивного типа в параметр метода, ожидающего соответствующий ему класс-обёртку.
- При присвоении значения примитивного типа переменной соответствующего класса-обёртки.

```
Character ch = 'a';
Integer i1 = 220;
Double d1 = 300.0;
Boolean b1 = false;
```

Автораспаковка
unboxing

Автораспаковка

Распаковка (unboxing) — конвертация класса-обёртки в соответствующий ему примитивный тип.

В процессе распаковки может произойти исключение java.lang.NullPointerException , если значение переменной равно null.).

Автораспаковка

```
class Main {

    public static void method1(int x) {
    }

    public static void main(String[] args) {
        Integer i1 = 100;

        method1(i1); // распаковка

        Double d1 = 2.3;
        Double d2 = 3.3;
        double d3 = i1 + d1 + d2; // распаковка

        System.out.println(d3);
    }
}
```

Автораспаковка

Компилятор Java автоматически применяет распаковку в следующих случаях:

- При передаче объекта класса-обёртки в метод, ожидающий соответствующий примитивный тип.
- При присвоении экземпляра класса-обёртки переменной соответствующего примитивного типа.
- В выражениях, в которых один или оба аргумента являются экземплярами классов-обёрток (кроме операции == и !=).

Пример

```
Integer x = 10;

Integer y = 20;


System.out.println("x > y : " + (x > y)); // false

System.out.println("x < y : " + (x < y)); // true
```

Пример

```
Integer x1 = new Integer(10);
Integer x2 = new Integer(10);

// x1 и x2 ссылаются на разные экземпляры объектов
System.out.println("x1 >= x2 : " + (x1 >= x2)); //true
System.out.println("x1 <= x2 : " + (x1 <= x2)); //true


//false происходит сравнение ссылок
System.out.println("x1 == x2 : " + (x1 == x2));

// true происходит сравнение ссылок
System.out.println("x1 != x2 : " + (x1 != x2));
```

Спасибо за внимание!