Plan

2023-10-09

1. Homework Review
2. Wrapper

3. String

1. Разбор домашнего задания
2. Wrapper
3. String

Theory

▶ **English**

▼ **На русском**

Wrapper

У примитивных типов есть объекты-аналоги - так называемые "классы оболочки", или "wrapper" (с англ. "обертка, упаковка"):

Primitive Data Types	Wrapper Classes
int	Integer
short	Short
long	Long
byte	Byte
float	Float
double	Double
char	Character
boolean	Boolean

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

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

```
public static void main(String[] args) {

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

```
    Double d = new Double(2.33);

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

Зачем они нужны?

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

- Например, обычный int занимает меньше места, и если нет необходимости проводить над ним особые операции
- с помощью класса-оболочки Integer можно выполнять специальные операции - например, перевести текст в число (с помощью метода .parseInt() для Integer-а). Если попробовать сделать это с помощью кода самому придется изрядно повозиться.

Примитивные типы потому и называют примитивными, потому что они лишены многих “тяжеловесных” особенностей объектов. Да, у объекта есть много удобных методов, но ведь они не всегда нужны.

Autoboxing/Autounboxing

Одной из особенностей примитивов и их классов-оберток в Java является автоупаковка/ автораспаковка (Autoboxing/Autounboxing)

Переменной класса-обертки можно присваивать значение примитивного типа. Этот процесс называется автоупаковкой (autoboxing).

Точно так же переменной примитивного типа можно присваивать объект класса-обертки. Этот процесс называется автораспаковкой (autounboxing).

```
public class Main {
    public static void main(String[] args) {
        int x = 7;
        Integer y = 111;
        x = y; // автораспаковка
        y = x * 123; // автоупаковка
    }
}
```

Методы классов оболочек

Методы valueOf()

Иногда в объекте типа String содержится число, и Вам нужно с ним работать дальше Метод valueOf() предоставляет второй способ создания объектов оболочек. Метод

перегруженный, для каждого класса существует два варианта - один принимает на вход значение соответствующего типа, а второй - значение типа String. Так же как и с конструкторами, передаваемая строка должна содержать числовое значение. Исключение составляет опять же класс Character - в нем объявлен только один метод, принимающий на вход значение char.

И в целочисленные классы Byte, Short, Integer, Long добавлен еще один метод, в который можно передать строку, содержащую число в любой системе исчисления. Вторым параметром вы указываете саму систему исчисления.

```
public class WrapperValueOf {
    public static void main(String[] args) {
        Integer integer1 = Integer.valueOf("6");
        Integer integer2 = Integer.valueOf(6);


        System.out.println(integer1);
        System.out.println(integer2);
    }
}
```

Методы parseXxx()

В каждом классе оболочке содержатся методы, позволяющие преобразовывать строку в соответствующее примитивное значение. В классе Double - это метод parseDouble(), в классе Long - parseLong() и так далее. Разница с методом valueOf() состоит в том, что метод valueOf() возвращает объект, а parseXxx() - примитивное значение.

Также в целочисленные классы Byte, Short, Integer, Long добавлен метод, в который можно передать строку, содержащую число в любой системе исчисления. Вторым параметром вы указываете саму систему исчисления. Следующий пример показывает использование метода parseLong():

```
public class WrapperDemo3 {
    public static void main(String[] args) {
        Long long1 = Long.valueOf("45");
        long long2 = Long.parseLong("67");

        System.out.println("long1 = " + long1);
        System.out.println("long2 = " + long2);
    }
}
```

Методы toString()

Все типы оболочки переопределяют toString(). Этот метод возвращает читабельную для человека форму значения, содержащегося в оболочке. Это позволяет выводить значение, передавая объект оболочки типа методу println():

```
Double double1 = Double.valueOf("4.6");
System.out.println(double1);
```

Также все числовые оболочки типов предоставляют статический метод toString(), на вход которого передается примитивное значение. Метод возвращает значение String:

```
String string1 = Double.toString(3.14);
```

Integer и Long предоставляют третий вариант toString() метода, позволяющий представить число в любой системе исчисления. Он статический, первый аргумент – примитивный тип, второй - основание системы счисления:

```
String string2 = Long.toString(254, 16); // string2 = "fe"
```

Методы toHexString(), toOctalString(), toBinaryString()

Integer и Long позволяют преобразовывать числа из десятичной системы исчисления к шестнадцатеричной, восьмеричной и двоичной системе. Например:

```
public class WrapperToXString {
    public static void main(String[] args) {
        String string1 = Integer.toHexString(254);
        System.out.println("254 в 16-ой системе = " + string1);

        String string2 = Long.toOctalString(254);
        System.out.println("254 в  8-ой системе = " + string2);

        String string3 = Long.toBinaryString(254);
        System.out.println("254 в  2-ой системе = " + string3);
    }
}
```

В классы Double и Float добавлен только метод toHexString().

Методы isNaN() и isInfinite() в классах Double и Float

`isNaN()`

Метод `isNaN()` используется для определения, является ли значение объекта `Double` или `Float` "Not-a-Number" (NaN). В математических операциях и вычислениях "Not-a-Number" (NaN) является символом, который не соответствует ни одному числу с плавающей запятой.

Примеры, когда может возникнуть NaN:

- Деление 0.0 на 0.0
- Квадратный корень из отрицательного числа
- Преобразование бесконечности в целое число

```
double x=Math.sqrt(-10);
boolean checkNaN=Double.isNaN(x);  // checkNaN будет true
```

или

```
Double xObj=new Double(Math.sqrt(-10));
boolean checkNaN=xObj.isNaN();  // checkNaN будет true
```

`isInfinite()`

Метод `isInfinite()` проверяет, является ли значение объекта `Double` или `Float` бесконечным. В контексте чисел с плавающей точкой бесконечность представляется значениями `Infinity` и `-Infinity`, которые являются результатом операций, таких как деление положительного числа на ноль.

Примеры, когда может возникнуть бесконечность:

- Деление положительного числа на 0.0
- Умножение бесконечности на положительное число

```
double y = 1.0/0.0;
boolean checkInfinite = Double.isInfinite(y);  // checkInfinite будет true
```

или

```
Double yObj = new Double(1.0/0.0);
boolean checkInfinite = yObj.isInfinite();  // checkInfinite будет true
```

Что такое NaN (Not-a-Number)?

NaN - это особый случай числа с плавающей точкой, который означает, что математической операции не удалось вернуть нормальное и понятное число. Это может случиться, например, когда вы пытаетесь разделить 0 на 0 или извлечь квадратный корень из отрицательного числа. В таких случаях результат не определен в математическом смысле, и Java возвращает NaN.

Что такое бесконечность (Infinity)?

Когда вы делите положительное число на ноль, математически результат стремится к бесконечности. В Java это состояние представляется как Infinity или -Infinity (если число отрицательное).

Пример приведения типов

```
        Integer iOb = new Integer(1000);

        System.out.println(iOb.byteValue());    // byte
        System.out.println(iOb.shortValue());   // short
        System.out.println(iOb.intValue());     // int
        System.out.println(iOb.longValue());    // long
        System.out.println(iOb.floatValue());   // float
        System.out.println(iOb.doubleValue());  //double
```

Статические константы классов оболочек

Каждый класс оболочка содержит статические константы, содержащие максимальное и минимальное значения для данного типа.

Например, в классе Integer есть константы Integer.MIN*VALUE – минимальное int значение и Integer.MAX*VALUE – максимальное int значение.

Класс `String` в Java

Строки в Java представлены классом `String`. Этот класс предоставляет множество методов для различных операций со строками.

Создание строк

Строки можно создавать разными способами:

```
        String str1="Привет, мир!";
        String str2=new String("Привет, мир!");
```

Основные методы

- **`length()`**: Возвращает длину строки.

```
int len = str1.length();  // 12
```

- **`charAt(int index)`**: Возвращает символ строки по указанному индексу.

```
char ch = str1.charAt(0);  // 'П'
```

- **`substring(int beginIndex, int endIndex)`**: Возвращает подстроку, начиная с `beginIndex` и заканчивая `endIndex - 1`.

```
String sub = str1.substring(0, 6);  // "Привет"
```

- **`concat(String str)`**: Конкатенация (склеивание) строк.

```
String newStr = str1.concat(" Как дела?");  // "Привет, мир! Как дела?"
```

- **`indexOf(String str)`** и **`lastIndexOf(String str)`**: Возвращает индекс первого и последнего вхождения подстроки в строке соответственно.

```
int first = str1.indexOf('м');  // 8
int last = str1.lastIndexOf('т');  // 5
```

- **`replace(char oldChar, char newChar)`**: Заменяет все вхождения символа `oldChar` на `newChar`.

```
String replaced = str1.replace(' ', '_');  // "Привет,_мир!"
```

- **`toLowerCase()`** и **`toUpperCase()`**: Возвращает новую строку, где все символы преобразованы к нижнему или верхнему регистру.

```
String lower = str1.toLowerCase();  // "привет, мир!"
String upper = str1.toUpperCase();  // "ПРИВЕТ, МИР!"
```

- **`trim()`**: Удаляет пробелы в начале и в конце строки.

```
String trimmed = "  Привет, мир!  ".trim();  // "Привет, мир!"
```

- **`split(String regex)`**: Разбивает строку на массив подстрок, используя регулярное выражение.

```
String[] words = str1.split(" ");  // ["Привет,", "мир!"]
```

Это лишь некоторые из многочисленных методов, предоставляемых классом `String`. С их помощью можно эффективно манипулировать строками и производить различные операции.

Homework

▶ English

▼ На русском

Это дз переносится на следующий день

Требования:

- Для выполнения дз создайте отдельный класс с именем StringTasks и реализуйте статичные методы для каждой задачи.
- Создайте класс Application с методом public static void main(String[] args)
- Все методы класса StringTasks должны вызваться из класса Application

▶ **Пример**

Задача 1. Реализуйте метод, который подсчитает количество цифр в строке.

Пример 1: Дана строка "I am agent 007", Результат: В строке 3 цифр(ы)

Пример 2: Дана строка "In 2022, I went to the sea twice", Результат: В строке 4 цифр(ы)

Пример 3: Дана строка "I was in Berlin 3 times in 2023, and in 2022 I was there twice", Результат: В строке 9 цифр(ы)

- попробуйте разные подходы, с разбиением стоки на массив символов и с использованием charAt

Задача 2. Реализуйте метод, который подсчитает количество только верхнего регистра в строке от A до Z.

Пример 1: Дана строка "Hello World", Результат: 2 букв(ы) верхнего регистра

Пример 2: Дана строка "In 2022, I went to the sea twice", Результат: 2 букв(ы) верхнего регистра

Пример 3: Дана строка "I was in Berlin 3 times in 2023, and in 2022 I was there twice", Результат: 3 букв(ы) верхнего регистра

Задача 3*. Реализуйте метод, который принимает строку и возвращает новую строку, в которой все слова перевернуты, но порядок слов остается прежним.

Пример 1: Дана строка "Hello World", Результат: "olleH dlroW"

Пример 2: Дана строка "Java Programming", Результат: "avaJ gnimmargorP"

Пример 3: Дана строка "Easy come easy go", Результат: "ysaE emoc ysae og"

Code

Lesson_24/src/string/StringApp.java

```java
package string;

/**
 * @author Andrej Reutow
 * created on 09.10.2023
 */
public class StringApp {

    public static void main(String[] args) {
        String str1 = "Hello";
        char[] str1AsChar = {'H', 'e', 'l', 'l', 'o'};

        System.out.println("String length " + str1.length()); // узнать дл
        System.out.println("chars length:" + str1AsChar.length);

        String str2 = "\n\n\n\n";
        System.out.println(str2.length()); // 4

        String str3 = new String(str1AsChar);

        System.out.println(str1.equals(str3)); // true

        char[] str3CharArray = str3.toCharArray();
        System.out.println();

        // Hello olleH
        String str4 = StringTasks.reverse(str1); // olleH
        System.out.println(str4); // olleH

        String str5 = str4.toUpperCase();
        String str6 = str4.toLowerCase();

        System.out.println(str4); // olleH
        System.out.println(str5); // OLLEH
        System.out.println(str6); // olleh

        System.out.println(str4.toUpperCase());
    }
}
```

Lesson_24/src/string/StringTasks.java

```java
package string;

/**
 * @author Andrej Reutow
 * created on 09.10.2023
 */
public class StringTasks {

    public static String reverse(String value) { // Hello
        String result = "";
        char[] valueChars = value.toCharArray(); // H,e,l,l,o
        // 0 "H"
        // 4 ...
        // 5 "o"

        for (int i = valueChars.length - 1; i >= 0; i--) {
            result = result + valueChars[i];
            // 1. result = o
            // 2. result = o + l = ol
            // 3. result = ol + l = oll
            // 4. result = oll + e = olle
            // 5. result = olle + H = olleH
        }

        return result;
    }
}
```

Lesson_24/src/wrapper/WrappersApp.java

```java
package wrapper;

/**
 * @author Andrej Reutow
 * created on 09.10.2023
 */
public class WrappersApp {

    public static void main(String[] args) {
        int intNumber = 5;
```

```
Integer integerNumber1 = new Integer(10);
Integer integerNumber2 = new Integer(10);

System.out.println(integerNumber1.equals(intNumber)); // false
System.out.println(integerNumber1.equals(1000)); // false

System.out.println(integerNumber1 == integerNumber2); // false
System.out.println(integerNumber1.equals(integerNumber2)); // true

System.out.println(integerNumber2.toString());

String stringInteger = integerNumber2.toString();

System.out.println(5 + integerNumber1); // 15
System.out.println("5" + integerNumber1.toString()); // "5" + "10"
System.out.println(5 + (integerNumber1 + integerNumber2)); // 5 + (
System.out.println(5 + (integerNumber1 + integerNumber2.toString())
// (10 + "10") = "10" + "10" = "1010"
// 5 + "1010" = "5" + "1010" = "51010"
System.out.println(5 + (integerNumber1 + "world")); // 510world
System.out.println((intNumber + integerNumber1) + "world"); // 15wo
System.out.println((5 + integerNumber1.toString()) + "world"); ///

System.out.println("" + integerNumber1 + integerNumber2); // 1010
System.out.println(integerNumber1 + integerNumber2 + ""); // 20
System.out.println(integerNumber1 + "" + integerNumber2); // 1010


int convertedFromWrapperToInt = integerNumber2.intValue(); // расп
byte convertedFromWrapperToByte = integerNumber2.byteValue(); // pa
short convertedFromWrapperToShort = integerNumber2.shortValue(); //
double convertedFromWrapperToDouble = integerNumber2.doubleValue();

Double doubleWrapper = new Double(5.5);
Double doubleWrapperV2 = 5.5;
Integer integerV2 = 555;
Long longV2 = 565L;

Float floatWrapper = 6.87F;

Double result = doubleWrapper / 0;
```

```
        System.out.println(result);
        System.out.println(result.isInfinite());


//        Integer resInt = integerV2 / 0; // вызовит исключение "деление на
//        System.out.println(resInt);
        System.out.println("\nСтатические методы оболочек");
        System.out.println("int from " + Integer.MIN_VALUE + " to " + Integ
        System.out.println(Short.MAX_VALUE); // максимальное значение short
        System.out.println(Long.MAX_VALUE); // максимальное значение long
        System.out.println(Double.MAX_VALUE); // максимальное значение doub

        Short someShort;
        Long longValue = (long) Integer.MAX_VALUE;
        if (longValue <= Short.MAX_VALUE) {
            someShort = longValue.shortValue();
        }


        System.out.println("\nParse value from String to Integer");
        String someStr1 = "007";
        String someStr2 = "100";
        Integer res; // 107

        Integer parsedInt1 = Integer.parseInt(someStr1); // 7
        Integer parsedInt2 = Integer.parseInt(someStr2); // 100
        res = parsedInt1 + parsedInt2; // 107
        System.out.println(res);

        Double.parseDouble("5.5");
        Long.parseLong("5");
        Short.parseShort("5");


//        Integer parsedIntInvalid1 = Integer.parseInt("someNumber 007"); //
        // так как срока "someNumber 007" содержит не только цифры
//        Integer parsedIntInvalid2 = Integer.parseInt("7 7"); // NumberFor

        int primitiveInt = 5;
        Integer integer = Integer.valueOf(primitiveInt); // 5
        Integer integer2 = primitiveInt; // 5
        Integer integer3 = Integer.valueOf("2023"); // 2023
```

```
        }
    }
```