Objektovo orientované programovanie

(obálkové triedy)

8. prednáška (2.časť)

Vladislav Novák
FEI STU v Bratislave
4.11.2014

Obsah

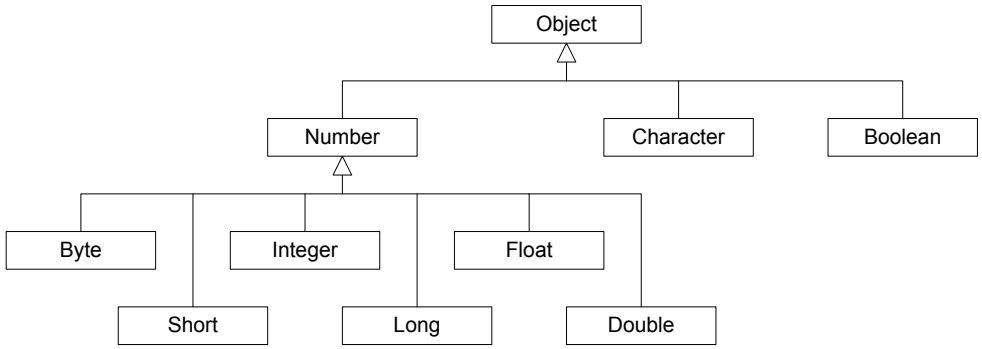
Obálkové triedy (wrapper-y)

Java poskytuje pre všetky základné typy (`int`, `double`, `boolean`, `char`, ...) obálkové triedy. *Obálková trieda* (*wrapper*) poskytuje obal pre základné typy, ktorý je vhodné použiť v prípadoch, keď potrebujeme reprezentovať napr. číslo, znak, alebo logickú hodnotu ako objekt.

Obálkové triedy:

- pre čísla:
  - `Byte` (pre typ `byte`)
  - `Short` (pre typ `short`)
  - `Integer` (pre typ `int`)
  - `Long` (pre typ `long`)
  - `Float` (pre typ `float`)
  - `Double` (pre typ `double`)
- pre znaky
  - `Character` (pre typ `char`)
- pre boolean
  - `Boolean` (pre typ `boolean`)

Obálkové triedy pre čísla sú podtriedami triedy `Number`:



Poznámka: Existujú aj iné triedy, napr.:

- `BigInteger, BigDecimal` – (podtriedy `Number`) pre výpočty s vysokou presnosťou
- `AtomicInteger, AtomicLong` – (podtriedy `Number`) pre aplikácie s viacerými vláknami
- `AtomicBoolean` – pre aplikácie s viacerými vláknami

Ak použijeme základný typ tam, kde sa očakáva objekt, kompilátor často zaistí *automatické zabalenie* (*auto-boxing*) základného typu do jeho obálkovej triedy. Podobne v prípade použitia objektu, na mieste kde sa predpokladá použitie základného typu, kompilátor *automaticky rozbalí* obálkovú triedu (*auto-unboxing*).

príklad automatického zabalenia a rozbalenia čísiel:

```
public class AutomatickeZabalenieARozbalenieCislic {
    public static void main(String[] args) {
        Integer a;
        Integer b;
        Integer c;
        int d;

        a = 12; //automaticke zabalenie cisla 12
        b = 15; //automaticke zabalenie cisla 15
        d = a;  //automaticke rozbalenie
        c = d;  //automaticke zabalenie
        c = a + b; //automaticke rozbalenie pred scitanim,
                 //automaticke zabalenie pred ulozenim vysledku do c
        System.out.println(c);

        medota1(a); //automaticke rozbalenie
        metoda2(d); //automaticke zabalenie
        metoda2(15);//automaticke zabalenie
        Integer p = metoda3(); //automaticke zabalenie
        int q = metoda4();     //automaticke rozbalenie
    }

    private static void metoda1(int parameter) {
        System.out.println(parameter);
    }

    private static void metoda2(Integer parameter) {
        System.out.println(parameter);
    }

    private static int metoda3() {
        return 100;
    }

    private static Integer metoda4() {
        return new Integer(200);
    }
}
```

príklad automatického zabalenia a rozbalenia znakov:

```
public class AutomatickeZabalenieARozbalenieZnakov {
     public static void main(String[] args) {
        Character a;
        char b;

        a = 'a'; //automaticke zabalenie
        b = a;   //automaticke rozbalenie
        a = b;   //automaticke zabalenie

        metoda1(a);  //automaticke rozbalenie
        metoda2('a');//automaticke zabalenie
        metoda2(b);  //automaticke zabalenie
    }
    private static void metoda1(char parameter) {
        System.out.println(parameter);
    }
    private static void metoda2(Character parameter) {
        System.out.println(parameter);
    }
}
```

príklad automatického zabalenia a rozbalenie logických hodnôt

```
public class AutomatickeZabalenieARozbalenieBoolean {
    public static void main(String[] args) {
        Boolean a;
        boolean b;

        a = true; //automaticke zabalenie
        b = a;    //automaticke rozbalenie
        a = b;    //automaticke zabalenie
    }
}
```

Čísla

Trieda `Number`

Metódy vracajúce hodnotu objektu ako príslušný základný typ:

```
byte      byteValue()
double    doubleValue()
float     floatValue()
int       intValue()
long      longValue()
short     shortValue()
```

Tieto metódy sú implementované všetkými podtriedami triedy `Number`.

Trieda `Integer`

Niektoré konštanty:

`static int MAX_VALUE`
maximálna hodnota typu `int` 2^{31}-1.

`static int MIN_VALUE`
minimálna hodnota typu `int` -2^{31}.

`static int SIZE`
počet bitov použitých na reprezentáciu hodnoty typu `int` v dvojkovom doplnku

`static Class<Integer> TYPE`
inštancia triedy `Class` reprezentujúca primitívny typ `int` (`int.class`)

Konštruktory:

`Integer``(int value)`
hodnota reprezentovaná novou inštanciou je rovnaká ako parameter `value`

`Integer``(String s)`
hodnota reprezentovaná novou inštanciou je daná vstupným reťazcom

Niektoré metódy:

`int` **`compareTo`**`(Integer anotherInteger)`
Numerické porovnanie dvoch inštancii triedy `Integer`. Návratová hodnota je 0, ak sú hodnoty rovnaké, menšia ako nula ak je Integer numericky menší ako argument metódy, väčšia ako nula ak je Integer numericky väčší ako argument metódy

`boolean` **`equals`**`(Object obj)`
Vráti `true`, ak sú hodnoty rovnaké, inak vráti `false`.

`static Integer` **`decode`**`(String nm) throws NumberFormatException`
Vráti inštanciu typu `Integer`, ktorej hodnota je daná vstupným reťazcom. Vstupný reťazec môže byť v desiatkovej, šestnástkovej, alebo osmičkovej sústave

`static Integer` **`valueOf`**`(int i)`
Vráti inštanciu typu `Integer`, ktorej hodnota je daná vstupným parametrom.

`static Integer` **`valueOf`**`(String s) throws NumberFormatException`
Vráti inštanciu typu `Integer`, ktorej hodnota je daná vstupným parametrom. Vstupný reťazec môže obsahovať iba číslo v desiatkovej sústave

`static Integer` **`valueOf`**`(String s, int radix) throws NumberFormatException`
Vráti inštanciu typu `Integer`, ktorej hodnota je daná vstupnými parametrami. Parameter `radix` určuje číselnú sústavu, v ktorej je zapísané číslo v reťazci `s`.

`static int` **`parseInt`**`(String s) throws NumberFormatException`
Vráti hodnotu typu `int` reprezentovanú vstupný parametrom. Vstupný parameter obsahuje číslo v desiatkovej sústave

`static int` **`parseInt`**`(String s, int radix) throws NumberFormatException`
Vráti hodnotu typu `int` reprezentovanú vstupnými parametrami. Parameter `s` určuje hodnotu, parameter `radix` určuje číselnú sústavu použitú v reťazci `s`.

`String` **`toString`**`()`
Vráti hodnotu inštancie ako typ `String`

`static String` **`toString`**`(int i)`
Vráti `String` obsahujúci hodnotu danú vstupným parametrom
`static String` **`toString`**`(int i, int radix)`
Vráti `String` obsahujúci hodnotu danú vstupnými parametrami. Parameter `i` určuje hodnotu, parameter `radix` určuje číselnú sústavu.

Ďalšie metódy slúžia napríklad na vykonávanie bitových operácii, vrátenie reťazca reprezentujúceho hodnotu v šestnástkovej, alebo osmičkovej sústave.

príklad

```
public static void main(String[] args) {
    Integer a = 10;
    Integer b = 20;

    System.out.println(a.compareTo(b)); //-1
    System.out.println(a.equals(b)); //false

    System.out.println(Integer.valueOf("245")); //245
    System.out.println(Integer.valueOf("F5",16)); //245
    System.out.println(Integer.valueOf("f5",16)); //245

    System.out.println(Integer.toString(245,16)); //f5
    System.out.println(Integer.toHexString(245)); //f5
    System.out.println(Integer.toOctalString(245)); //365
    System.out.println(Integer.toBinaryString(245));//11110101

    System.out.println(Integer.decode("0xF5")); //245
    System.out.println(Integer.decode("0XF5")); //245
    System.out.println(Integer.decode("#F5")); //245
    System.out.println(Integer.decode("0365")); //245

    System.out.println(Integer.parseInt("245")); //245
    System.out.println(Integer.parseInt("F5", 16)); //245

    System.out.println(Integer.signum(10)); //1
    System.out.println(Integer.signum(0)); //0
    System.out.println(Integer.signum(-10)); //-1

    System.out.println(Integer.rotateLeft(10, 2)); //40 = 10*2^2

    System.out.println(new Integer(10).doubleValue()); //10.0

    System.out.println(Integer.MAX_VALUE); //2147483647
    System.out.println(Integer.SIZE); //32
}
```

Ostatné obálkové číselné triedy majú podobné metódy.

Znaky

Obálková trieda pre znaky je trieda `Character`
Inštancie triedy `Character` sú nemenné (immutable). To znamená, že po vytvorení objektu nemožno meniť jeho stav.

Niektoré metódy triedy `Character`:

Useful Methods in the `Character` Class

Method	Description
`boolean isLetter(char ch)` `boolean isDigit(char ch)`	určí, či je znak písmenom určí, či je znak číslicou
`boolean isWhitespace(char ch)`	určí, či je znak bielim znakom
`boolean isUpperCase(char ch)` `boolean isLowerCase(char ch)`	určí, či je znak veľkým písmenom určí, či je znak malým písmenom
`char toUpperCase(char ch)` `char toLowerCase(char ch)`	vráti ten istý znak ako veľké písmeno vráti ten istý znak ako malé písmeno
`toString(char ch)`	Vráti reťazec reprezentujúci vstupný znak

príklad:

```
public static void main(String[] args) {
    System.out.println(Character.isLetter('a')); //true
    System.out.println(Character.isLetter('1')); //false
    System.out.println(Character.isLetter('+')); //false

    System.out.println(Character.isDigit('a')); //false
    System.out.println(Character.isDigit('1')); //true

    System.out.println(Character.isUpperCase('a')); //false
    System.out.println(Character.isUpperCase('A')); //true

    System.out.println(Character.toUpperCase('a')); //A
}
```

Escape Sequences	
Escape Sequence	**Description**
`\t`	Insert a tab in the text at this point.
`\b`	Insert a backspace in the text at this point.
`\n`	Insert a newline in the text at this point.
`\r`	Insert a carriage return in the text at this point.
`\f`	Insert a formfeed in the text at this point.
`\'`	Insert a single quote character in the text at this point.
`\"`	Insert a double quote character in the text at this point.
`\\`	Insert a backslash character in the text at this point.

príklad:

```
public static void main(String[] args) {
    System.out.println("jeden \\ dva \"tri\' styri\npat");
}
```

výstup:

```
jeden \ dva "tri' styri
pat
```