Datové typy v Javě

Tomáš Pitner, upravil Marek Šabo

Úvod k datovým typům v Javě

- Existují dvě základní kategorie datových typů: **primitivní** a **objektové**

Primitivní

- v proměnné je uložena přímo hodnota
- např. `int`, `long`, `double`, `boolean`, …

Objektové

- musí se nejdřív zkonstruovat (použitím `new`)
- do proměnné se uloží pouze odkaz
- např. `String`, `Person`, …

Datové typy primitivní

integrální typy

zahrnují typy *celočíselné* (byte , short , int a long) a typ char

čísel s pohyblivou řádovou čárkou

float a double

logických hodnot

boolean

Výchozí (default) hodnoty

- Každý typ má svou výchozí (default) hodnotu, na kterou je nastaven, není-li hned přiřazena jiná.
- Dle Java Language Specification: Each *class variable*, *instance variable*, or array component is initialized with a default value when it is created (§15.9, §15.10):
  - For type `byte`, the default value is zero, that is, the value of `(byte)0`.
  - For type `short`, the default value is zero, that is, the value of `(short)0`.
  - For type `int`, the default value is zero, that is, `0`.
  - For type `long`, the default value is zero, that is, `0L`.
  - For type `float`, the default value is positive zero, that is, `0.0f`.
  - For type `double`, the default value is positive zero, that is, `0.0d`.
  - For type `char`, the default value is the null character, that is, `'\u0000'`.
  - For type `boolean`, the default value is `false`.
  - For all reference types (§4.3), the default value is `null`

Na co se vztahují výchozí hodnoty

- Automatické nastavení proměnných na výchozí hodnoty se tedy vztahuje na proměnné objektů a tříd (atributy) a prvky polí.
- Nevztahuje se na lokální proměnné a parametry, ty musejí být před prvním použitím nastaveny, inicializovány.

Příklad výchozí hodnoty

```
int i; // automatically i = 0
```

Více informací najdete na: The Java Tutorials: Primitive Data Types.

Zajímavosti a odlišnosti

- V Javě neexistuje možnost typy rozsahově či interpretačně modifikovat (žádné unsigned int apod.)
- Pro velká čísla lze v nových verzích Javy použít notaci s podtržítkem k oddělení řádů po tisících:

```
private int bigNumber = 123_456_789;
```

Datové typy objektové

- objektovými typy v Javě jsou všechna ostatní typy
- **třídy**
- **rozhraní** ("téměř totéž, co třídy")
- **pole** (ano, v Javě jsou samotná pole objekty)

Výchozí hodnota objektového typu je `null` — tzv. "ukazatel na nic".

Příklad použití objektového typu

```
Person p; // p is null automatically
p = new Person(); // now p references to an object
```

- Objektový typ je všechno, kde se používá operátor `new`.

Shrnutí

Základní rozdíl je v práci s proměnnými.

primitivní typy
: přímo obsahují danou *hodnotu*

objektové typy
: obsahují pouze *odkaz* na příslušný objekt

- Důsledek: dvě objektové proměnné mohou nést odkaz na **tentýž objekt**

Přiřazení primitivní proměnné

- Hodnota proměnné se nakopíruje:

```
double a = 1.23456;
double b = a;
a += 2;
// a is 3.23456
// b is 1.23456
```

Přiřazení objektové proměnné

- Objektové proměnné ukazují na stejný objekt:

```
public class Counter {
   private double value;
   public Counter(double v) {
      value = v;
   }
   public void add(double v) {
      value += v;
   }
}
...
Counter c1 = new Counter(1.23456);
Counter c2 = c1;
c1.add(2);
// c1 has value 3.23456
// c2 has value 3.23456
```

Operátor ==

- Pro primitivní typy porovnává hodnoty:

```
1 == 1 // true
1 == 2 // false
```

- Pro objektové typy porovnává odkazy:

```
Counter c1 = new Counter(1.23456);
Counter c2 = c1;
c1 == c2 // true
c1 == new Counter(1.23456) // false
```

- Na porovnání *hodnot* objektových typů se používá `equals`, probereme později.

Pole v zkratce

- Vytvoření, naplnění a získání hodnot vypadá následovně:

```
int[] array = new int[2];
array[0] = 1;
array[1] = 4;
System.out.println("First element is: " + array[0]);
```

- Deklarace: *typ*[] *jméno* = `new` *typ* [*velikost*];
- *typ* může být i objektový: `Person[] p = new Person[3];`

Velikost pole

- *velikost* pole je daná při jejím vytvoření a **nelze ji měnit**
- V budoucnu budeme probírat *kolekce* (seznam, slovník), což je mocnější složený datový typ než pole
- jejich počty prvků se mohou dynamicky měnit

Mělká kopie

- Přiřazení proměnné objektového typu (což je i pole) vede pouze k **duplikaci odkazu**, nikoli celého odkazovaného objektu.
- Modifikace jedné proměnné se pak projeví u i té druhé.

```
int[] array = new int[] {1, 4, 7};
int[] array2 = array;
array[1] = 100;
System.out.println(array[1]); // prints 100
System.out.println(array2[1]); // prints 100
```

- Takovému kopírování se říká **mělká kopie** (*shallow copy*).

Hluboká kopie

Provedeme-li vytvoření nového pole, pak array2 obsahuje kopii (duplikát) původního pole.

```
int[] array = new int[] {1, 4, 7};
int[] array2 = Arrays.copyOf(array, array.length);
array[1] = 100;
System.out.println(array[1]); // prints 100
System.out.println(array2[1]); // prints 4
```

- Takovému kopírování se říká **hluboká kopie** (*deep copy*).

Metoda `copyOf` bere dva parametry — původní pole a počet prvků, kolik se má nakopírovat.

Hluboká kopie u objektů I

- Obdobně to funguje i objektů.

```
Person[] people = new Person[] { new Person("Jan"), new Person("Adam")};
Person[] people2 = Arrays.copyOf(people, people.length);
people[1] = new Person("Pepa");
System.out.println(people[1].getName()); // prints Pepa
System.out.println(people2[1].getName()); // prints Adam
```

- Co kdybychom změnili jenom *jméno* (atribut objektu)?

Hluboká kopie u objektů II

- Do cílového pole se zduplikují jenom *odkazy na objekty* Person, nevytvoří se kopie objektů Person!

```
Person[] people = new Person[] { new Person("Jan"), new Person("Adam")};
Person[] people2 = Arrays.copyOf(people, people.length);
people[1].setName("Pepa"); // changes Adam to Pepa
System.out.println(people[1].getName()); // prints Pepa
System.out.println(people2[1].getName()); // prints Pepa
```

- Jinými slovy, pole mají sice *různý odkaz* (šipku), ale na **stejný objekt**.
- V předešlém příkladu jsme změnili odkaz na jiný objekt.
- Teď jsme změnili obsah objektu, na který ukazují oba odkazy.