Konstruktory

Objektově orientované programování

Karel Šimerda

Fakulta elektrotechniky a informatiky

1. října 2019, Pardubice

Co se naučíme o konstruktorech

- Podrobnosti o práci konstruktorů [1]
  - co dělá statický
  - a co instanční
- Podle jakých pravidel se konstruktory chovají
- Z čeho se skládají
- Jaké jsou jejich funkce
- K čemu se používají

[1] Zpracováno podle zdroje:
PECINOVSKÝ, Rudolf. Java 8: úvod do objektové architektury pro mírně pokročilé. Praha: Grada Publishing, 2014. Knihovna programátora (Grada). ISBN 978-80-247-4638-8. Strana 456-472.

Zavádění tříd

- Třída je v Java objekt
  - je to zvláštní objekt
  - není totiž tvořen podle jiné třídy
  - má svůj seznam atributů a metod, které nejsou podle jiné třídy
- Třídy se zavádějí do paměti až když jsou zapotřebí
- Třídy jsou po překladu zapisovány do samostatných souborů s příponou `class`
- Třídy se natahují do paměti zavaděčem
  - některým potomkem třídy `java.lang.ClassLoader`
- Po natažení třídy do paměti se objekt třídy inicializuje statickým konstruktorem

Popis funkce statického konstruktoru

- Je to konstruktor objektu třídy
- Použije se pouze jednou, při zavedení třídy
- Objekty třídy nejsou instancemi nějaké třídy
- Třída má pouze jeden statický bezparametrický konstruktor
- Statický konstruktor se skládá z
  1. inicializačních příkazů v deklaraci statických atributů
  2. statických inicializačních bloků

Vlastnosti statického inicializačního bloku

- Používá se někdy jen označení „statický blok“
- Vznikne na místě, kde se smí deklarovat atribut nebo metoda
- Lze ho míchat s deklaracemi statických atributů
- Vytváří se jako běžný blok
- Před blokem musí být klíčové slovo `static`
- V bloku může být i složitější kód
- Statický blok je bezparametrická metoda na jedno použití
- V statickém bloku lze nastavovat hodnoty statických konstant
- Třída může obsahovat několik statických bloků
- Statické bloky se vykonávají v pořadí, jak jsou ve třídě napsány
- Více statických bloků ve třídě není dobrá praktika
- Více bloků se používá při ladění konstrukce třídy
  - kdy v blocích lze používat zarážky (breakpoints)
  - a lze prokládat deklarace s inicializaci statických atributů statickými bloky

```
1  public class StatickyKonstruktor {
2    static {
3      System.out.println("Zacatek statickeho bloku");
4    }
5    static final int MAX;
6    static final int MIN = 0;
7    static {
8    //    System.out.println("Pred inicializaci konstanty MAX="+MAX);
9    }
10   static {
11     MAX = 10;
12   }
13   static {
14     System.out.println("Po inicializaci konstanty MAX=" + MAX);
15   }
16   static {
17     // MIN = 0;
18   }
19   static {
20     System.out.println("Po inicializaci konstant MAX/MIN="+MIN+MAX);
21   }
22 }
```

Chování konstruktoru instancí

- Konstrukce instance objektu probíhá ve dvou fázích
  1. alokace paměti a její předběžná inicializace
  2. závěrečná inicializace vlastním kódem konstruktoru
- Přímé volání konstruktoru
  1. lze spustit po vykonání operátoru `new`
     - se vyhradí paměť
     - v které se nastaví odkaz na objekt třídy
     - a zbytek paměti se vynuluje
  2. nebo na počátku jiného konstruktoru
     - následně volanému konstruktoru se předá skrytý parametr `this`
     - a nechá se konstruktor pracovat nebo ten deleguje odpovědnost na další konstruktor atd.
     - teprve poslední delegovaný konstruktor vyhradí paměť a začne ji inicializovat
     - na závěr všechny ostatní delegované konstruktory i sám prvně volaný konstruktor postupně dokončí inicializaci objektu

Instanční inicializační blok

- Slouží k inicializaci každé instance
- Od statického bloku se liší tím, že nemá před blokem `static`
- Instanční bloky a instanční deklarace jsou spuštěny v pořadí, jak jsou zapsány ve třídě
- V inicializačních deklaracích a instančních blocích nelze používat atributy, které jsou deklarovány až za daným příkazem
- Instančních bloků může být v jedné třídě několik
- Lze je míchat s deklaracemi instančních atributů
- Všechny se vykonávají se před vstupem do těla konstruktoru
- Instanční bloky jsou zbytečné, protože to samé lze udělat v instančních konstruktorech
- Instanční bloku jsou užitečné při ladění chyb v inicializaci instančních proměnných

Dvě těla konstruktoru instancí

Konstruktor instancí se skládá ze dvou částí

1. z kódu tvořeného inicializačními deklaracemi a instančními inicializačními bloky
2. navazujícího kódu tvořeného vlastním tělem konstruktoru

V případě, konstruktory delegují konstrukci na jiné konstruktory

1. tak zpracovávají pouze parametry
2. teprve před posledním konstruktorem se spustí veškerá inicializace instančních atributů

Ukázka instančních inicializačních bloků

```
1  public class InstancniBlok {
2    {
3    System.out.println("Zacatek inicializace instance");
4    }
5    int prvniAtribut;
6    { System.out.println("Atribut prvni po deklaraci: " + prvniAtribut);
7      prvniAtribut = 1;
8    }
9    final int druhyAtribut;
10   final int tretiAtribut;
11   { System.out.println("Atribut prvni po zmene: " + prvniAtribut);
12     //System.out.println("Atribut druhy po deklaraci: " + druhyAtribut)
           ;
13   }
14   {
15    druhyAtribut = 2;
16   }
17   public InstancniBlok() {
18     this.tretiAtribut = 3;
19     System.out.println("Hodnoty atributu"
20     + prvniAtribut + "," + druhyAtribut + ","+ tretiAtribut + "."
21   );
```

Statický atribut s odkazem na instanci vlastní třídy

- Například při definici jedináčka (singleton)
- Musí se zabezpečit
  - inicializaci statických atributů
  - před spuštěním konstruktoru instance
  - a jeho volaných metod
  - které budou používat tyto statické atributy

Jedináček (Singleton) - Ukázka návrhového vzoru

```
1  public class StaticBlockSingleton {
2     private static final StaticBlockSingleton INSTANCE;
3
4     static {
5       try {
6         INSTANCE = new StaticBlockSingleton();
7       } catch (Exception e) {
8          throw new RuntimeException(e);
9       }
10    }
11
12    public static StaticBlockSingleton getInstance() {
13      return INSTANCE;
14    }
15
16    private StaticBlockSingleton() {
17      // ...
18    }
19 }
```

Jedináček (Singleton) - Ukázka návrhového vzoru

```java
public class ClassicSingleton {

  private static ClassicSingleton instance = null;

  private ClassicSingleton() {
    // Jen z duvodu, aby se mohl zavolat jen statickou metodou tridy.
  }
  public static ClassicSingleton getInstance() {
    if(instance == null) {
      instance = new ClassicSingleton();
    }
  return instance;
  }
}
```

```java
public class NewSingleton {
  private static final NewSingleton INSTANCE = new NewSingleton();
  public static NewSingleton getInstance()  {
    return INSTANCE;
  }
  private NewSingleton() {
  }
```

Ukázka jedináčka pro vícevláknové aplikace

```
1  public class Singleton {
2    private static volatile Singleton instance = null;
3
4    public static Singleton getInstance() {
5      if (instance == null)
6        synchronized (Singleton.class) {
7          if (instance == null)
8            instance = new Singleton();
9        }
10     return instance;
11   }
12 }
```

Děkuji za pozornost!