Seminář Java

Zavádění tříd

Radek Kočí

Fakulta informačních technologií VUT

Duben 2008

Zavádění tříd

JVM zavádí třídy dynamicky

Class loader

- objekt schopný zavádět třídy
- abstraktní třída **`java.lang.ClassLoader`**
- každá třída (**`java.lang.Class`**) obsahuje referenci na svůj class loader

Implementace class loaderu

- zajištění specifického chování při zavádění tříd
- více verzí stejné třídy

Zavádění tříd

Spuštění aplikace:
`java Test`

Kroky při spouštění:

- JVM zjistí, že třída **`Test`** není zavedena
- JVM použije zavaděč (class loader) pro zavedení třídy **`Test`**

Proces zavádění třídy:

- načtení byte-code (loading)
- linkování (linking)
- inicializace (initialization)

Zavádění tříd

Loading

- vyhledání a načtení byte-code třídy
- různé zavaděče mají různou politiku (souborový systém, http, ...)

Linking

- operace nutné pro to, aby byla třída použitelná
  - verification
    - verifikace korektnosti binární reprezentace třídy
  - preparation
    - vytváří statické členy třídy, implicitní inicializace
  - resolution (optional)
    - závislosti na jiných třídách

Zavádění tříd

Initialization

- explicitní inicializace statických členů třídy
- před inicializací se musí inicializovat nadřazená třída (pokud ještě není)
- k inicializaci třídy/rozhraní **T** dochází před akcí:
  - **T** je třída a instance **T** je vytvářena
  - **T** je třída a její statická metoda je volána
  - statická proměnna deklarovaná v **T** je přiřazena
  - statická proměnna deklarovaná v **T** je použita

Zavádění tříd

```
class Super {
    static { System.out.print("Super "); }
}
class One {
    static { System.out.print("One "); }
}
class Two extends Super {
    static { System.out.print("Two "); }
}
class Test {
    public static void main(String[] args) {
        One o = null;
        Two t = new Two();
        System.out.println((Object)o == (Object)t);
    }
}
```

$\Rightarrow$ **`Super Two false`**

Třída **`One`** nebude nikdy linkována (není aktivně použita).

Zavádění tříd

```
class Super { static int taxi = 1729; }

class Sub extends Super {
    static { System.out.print("Sub "); }
}

class Test {
    public static void main(String[] args) {
        System.out.println(Sub.taxi);
    }
}
```

1729 (Sub neni inicializovana)

Zavádění tříd

```
class Super { static int taxi = 1729; }

class Sub extends Super {
    static { System.out.print("Sub "); }
}

class Test {
    public static void main(String[] args) {
        System.out.println(Sub.taxi);
    }
}
```

1729 (Sub neni inicializovana)

Zavádění tříd

```
class Super { static int taxi = 1729; }

class Sub extends Super {
    static { System.out.print("Sub "); }
}

class Test {
    public static void main(String[] args) {
        Sub s = new Sub();
        System.out.println(Sub.taxi);
    }
}
```

Sub 1729

Zavádění tříd

```
class Super { static int taxi = 1729; }

class Sub extends Super {
    static { System.out.print("Sub "); }
}

class Test {
    public static void main(String[] args) {
        Sub s = new Sub();
        System.out.println(Sub.taxi);
    }
}
```

Sub 1729

Zavádění tříd

Zavaděče tříd (class loaders)

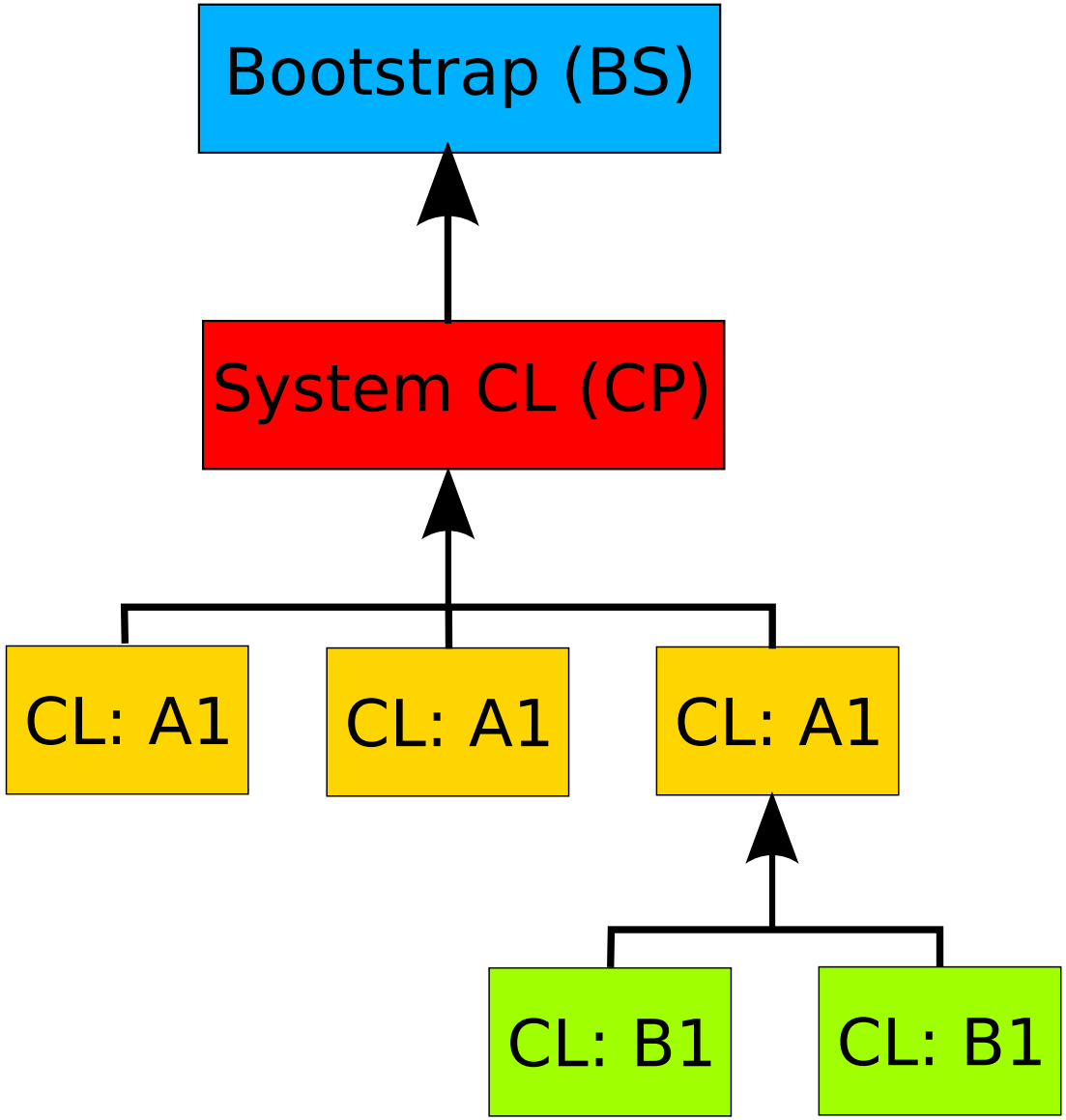
- vytvářejí hierarchickou strukturu
- každý zavaděč má svého předka (kromě vrcholového)
- zavaděč deleguje požadavky na svého předka
- každá třída je definována svým typem a svým zavaděčem
- každá třída je zavedena pouze jednou
- každá třída je uchována v zavaděči

Zavádění tříd – Hierarchie zavaděčů

Hierarchie zavaděčů

- **`bootstrap loader`**
  zavádí základní třídy JVM (**`jr.jar`**) + **`extensions`**
- **`system loader`**
  zavádí třídy dostupné přes **`CLASSPATH`** (nebo **`-classpath`**)

Hierarchie zavaděčů



Zavádění tříd

Metody třídy **`java.lang.ClassLoader`**

- **`loadClass()`**
  - test, zda už je třída zavedena (**`findLoadedClass`**)
  - deleguje zavádění (**`loadClass`**)
  - *fail* $\Rightarrow$ načte třídu (**`findClass`**)
- **`findClass()`**
  - čte byte-code reprezentující třídu
  - vytváří třídu (**`defineClass`**)
  - *odvozené zavaděče by měly předefinovat tuto metodu*
- **`defineClass()`**
  - jako parametr má pole bytů
  - konvertuje pole bytů na třídu
- **`resolveClass()`**
  - ověření symbolický referencí ze zaváděné třídy
  - zavedení závislých tříd (volitelné)

Zavádění tříd

```
protected Class findClass(String pClassName)
throws ClassNotFoundException {
   try {
      File lClassFile = ...
      InputStream lInput = ...
      ByteArrayOutputStream lOutput = ...
      int i = 0;
      while ((i = lInput.read()) >= 0) {
         lOutput.write( i );
      }
      byte[] lBs = lOutput.toByteArray();
      return defineClass(pClassName, lBs, 0,
                                    lBs.length);
   } catch (Exception e) {
      throw new ClassNotFoundException(...);
   }
}
```

Uvolnění třídy

Uvolnění objektu (*garbage collecting*)

- pokud již neexistuje reference na objekt z živého vlákna
- finalizace objektu

Třída může být uvolněna:

- pokud její zavaděč (class loader) podlehne **`garbage collectoru`**
- třídy zavedené *bootstrap* zavaděčem nemohou být nikdy uvolněny
- třída *nemůže* být uvolněna, pokud je její zavaděč potenciálně dostupný

Shrnutí

- Třídy jsou zaváděny dynamicky
- Každá třída má svůj zavaděč
- Existují implicitní zavaděče
- Každá třída ”dědí” zavaděč třídy, ze které bylo vyvoláno zavedení
- Je možné definovat vlastní zavaděče