12. Dokumentace a distribuce aplikací

- Dokumentace javových programů, dokumentace API
- Typy komentářů - dokumentační komentáře
- Generování dokumentace
- Značky javadoc
- Distribuční archívy .jar
- Vytvoření archívu, metainformace
- Spustitelné archívy

Dokumentace javových programů

Základním a standardním prostředkem je tzv. ***dokumentace API***

- Dokumentace je ***naprosto nezbytnou součástí*** javových programů.
- Rozlišujeme dokumentaci např. instalační, systémovou, uživatelskou, programátorskou...

Zde se budeme věnovat především dokumentaci programátorské, určené těm, kdo budou náš kód využívat ve svých programech, rozšiřovat jej, udržovat jej. Programátorské dokumentaci se říká ***dokumentace API*** (apidoc, apidocs).

Při jejím psaní dodržujeme tato pravidla:

- Dokumentujeme především ***veřejné*** (public) a ***chráněné*** (protected) prvky (metody, proměnné). Ostatní dle potřeby.
- Dokumentaci píšeme ***přímo do zdrojového kódu*** programu ***ve speciálních dokumentačních komentářích*** vpisovaných ***před příslušné prvky*** (metody, proměnné).
- Dovnitř metod píšeme jen ***pomocné komentáře*** pro programátory (nebo pro nás samotné).

Typy komentářů

Podobně jako např. v C/C++:

řádkové
: od značky // do konce řádku, nepromítnou se do dokumentace API

blokové
: začínají `/*` pak je text komentáře, končí */ na libovolném počtu řádků

dokumentační
: od značky /** po značku */ může být opět na libovolném počtu řádků

  Každý další řádek může začínat mezerami či *, hvězdička se v komentáři neprojeví.

Kde uvádíme dokumentační komentáře

Dokumentační komentáře uvádíme:

- Před ***hlavičkou třídy*** - pak komentuje třídu jako celek.
- Před ***hlavičkou metody nebo proměnné*** - pak komentuje

příslušnou metodu nebo proměnnou.
- Celý balík (package) je možné komentovat ***speciálním samostatným HTML souborem*** `package-summary.html` uloženým v adresáři balíku.

Generování dokumentace

Dokumentace má standardně podobu HTML stránek (s rámy i bez)

Dokumentace je generována nástrojem `javadoc` z

1. dokumentačních komentářů
2. i ze samotného zdrojového textu

Lze tedy (základním způsobem) dokumentovat i program bez vložených komentářů!

Chování `javadoc` můžeme změnit

1. volbami (options) při spuštění,
2. použitím jiného tzv. `doclet`u, což je třída implementující potřebné metody pro generování komentářů.

Princip generování ze zdrojových textů pomocí speciálních `doclet`ů se dnes používá i po jiné než dokumentační účely - např. pro generátory zdrojových kódu aplikací EJB apod.

Značky `javadoc`

`javadoc` můžeme podrobněji instruovat pomocí značek vkládaných do dokumentačních komentářů, např.:

@author
: specifikuje autora API/programu

@version
: označuje verzi API, např. "1.0"

@deprecated
: informuje, že prvek je zavrhovaný

@exception
: popisuje informace o výjimce, kterou metoda propouští ("vyhazuje")

@param
: popisuje jeden parametr metody

@since
: uvedeme, od kdy (od které verze pg.) je věc podporována/přítomna

@see
: uvedeme odkaz, kam je také doporučeno nahlédnout (související věci)

Příklad zdrojového textu se značkami javadoc

Zdrojový text třídy `Window`:

```
/**
 * Klasse, die ein Fenster auf dem Bildschirm repräsentiert
```

```
 * Konstruktor zum Beispiel:
 * <pre>
 *         Window win = new Window(parent);
 *        win.show();
 * </pre>
 *
 * @see                 awt.BaseWindow
 * @see                 awt.Button
 * @version         1.2 31 Jan 1995
 * @author          Bozo the Clown
**/
class Window extends BaseWindow
{
        ...
}
```

Příklad dokumentačního komentáře k proměnné:

```
    /**
     * enthält die aktuelle Anzahl der Elemente.
     * muss positiv und kleiner oder gleich der Kapazität sein
    **/
    protected int count;
```

Tyto a další příklady a odkazy lze vidět v původním materiálu JavaStyleGuide des IGE, odkud byly ukázky převzaty.

Spouštění `javadoc`

- **`javadoc`** `[options] [packagenames] [sourcefiles] [classnames] [@files]`
- možné volby:
  - `-help`, `-verbose`
  - `-public`, `-protected`, `-package`, `-private` - specifikuje, které prvky mají být v dokumentaci zahrnuty (implicitně: `-protected`)
  - `-d` ***`destinationdirectory`*** - kam se má dok. uložit
  - `-doctitle` ***`title`*** - titul celé dokumentace

Příklady

Zdroják s dokumentačními komentáři - Komentáře

Ukázkové spuštení `javadoc`

```
javadoc -classpath . -d apidocs svet
```

vytvoří dokumentaci tříd z balíku `svet` do adresáře `apidocs`

Distribuce aplikací

Distribucí nemyslíme použití nástroje typu "InstallShield"..., ale spíše něčeho podobného `tar`/`ZIP`u

- Java na sbalení množiny souborů zdrojových i přeložených (`.class`) nabízí nástroj `jar`.
- Sbalením vznikne soubor (archív) `.jar` formátově podobný `ZIP`u (obvykle ***je*** to ZIP formát), ale nemusí být komprimován.

- Kromě souborů obsahuje i metainformace (tzv. ***MANIFEST***)
- Součástí archívu nejsou jen .class soubory, ale i další zdroje, např. ***obrázky***, ***soubory s národními variantami řetězců***, ***zdrojové texty*** programu, ***dokumentace***...

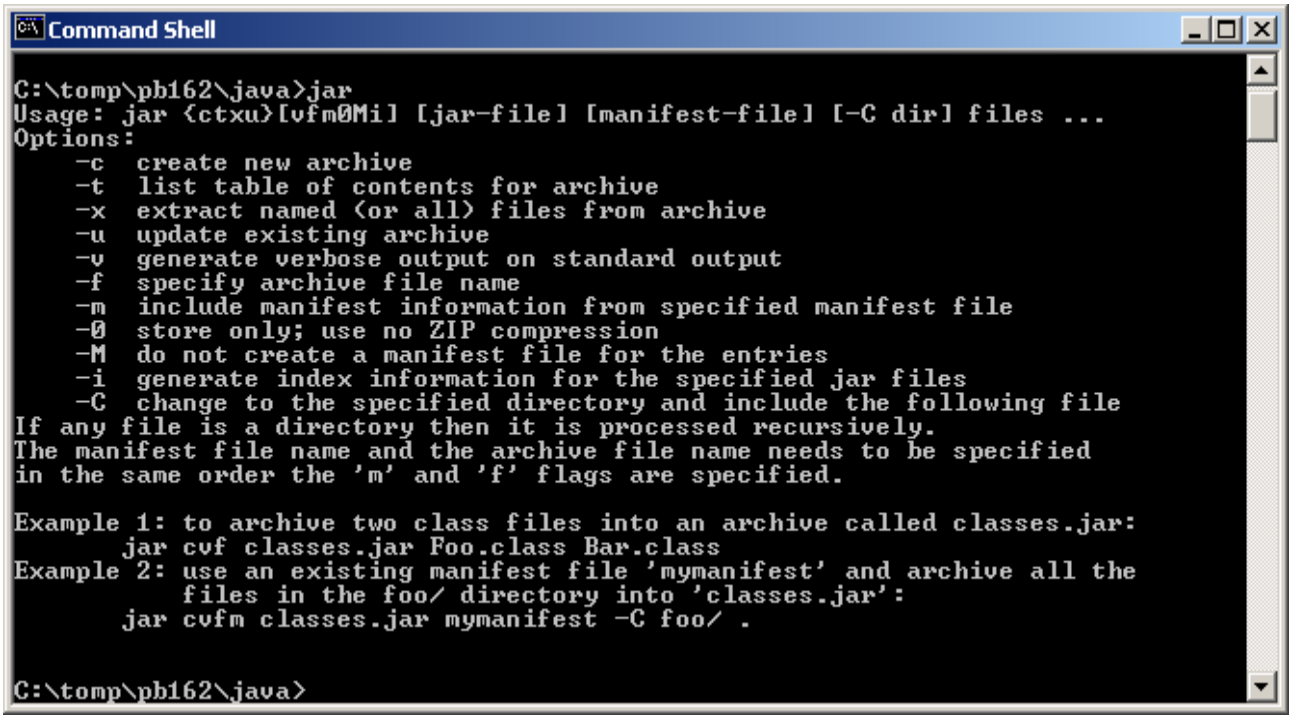
Spuštění `jar`

- **jar** `{ctxu} [vfm0M] [jar-file] [manifest-file] [-C dir] files`
  -
    - `c` - vytvoří archív
    - `t` - vypíše obsah archívu
    - `x` - extrahuje archív
    - `u` - aktualizuje obsah archívu
- volby:
    - `v` - verbose
    - `0` - soubory nekomprimuje
    - `f` - pracuje se se souborem, ne se "stdio"
    - `m` - přibalí metainformace z `manifest-file`
- parametr `files` uvádí, které soubory se sbalí - i nejavové (např. typicky dokumentace API - HTML, datové soubory)

Volby `jar`

Volby JAR lze vypsat i spuštěním jar bez parametrů:

Obrázek 11.1. Volby nástroje JAR



`jar` - příklad

Vezměme následující zdrojový text třídy `JarDemo` v balíku tomp.ucebnice.jar, tj. v adresáři `c:\tomp\pb162\java\tomp\ucebnice\jar`:

Obrázek 11.2. Třída JarDemo

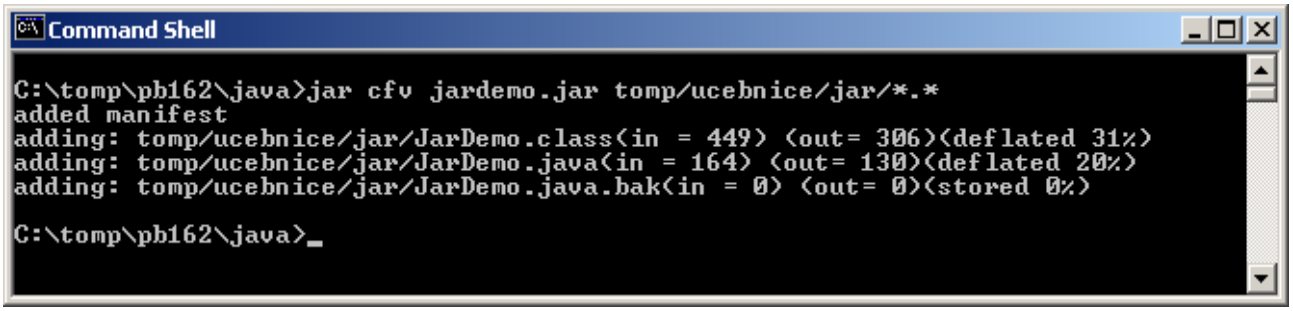
SortedMapComparatorDemo.java | ListIteratorDemo | SortedSetDemoOK.java | JarDer

```
package tomp.ucebnice.jar;

public class JarDemo {
    public static void main(String[] args) {
        System.out.println("Spustena trida JarDemo!");
    }
```

Vytvoříme archív se všemi soubory z podadresáře `tomp/ucebnice/jar` (s volbou c - create, v - verbose, f - do souboru):

Obrázek 11.3. Vytvoření archívu se všemi soubory z podadresáře tomp/ucebnice/jar



Vzniklý `.jar` soubor lze prohlédnout/rozbalit také běžným nástrojem typu unzip, gunzip, WinZip, PowerArchiver nebo souborovým managerem typu Servant Salamander...

Obrázek 11.4. .jar archiv v okně PowerArchiveru

Tento archív rozbalíme v adresáři `/temp` následujícím způsobem:

Obrázek 11.5. Vybalení všech souborů z archívu

Rozšíření `.jar` archívů

Formáty vycházející z `JAR`:

- pro webové aplikace - `.war`
- pro enterprise (EJB) aplikace - `.ear`

liší se podrobnějším předepsáním adresářové struktury a dalšími povinnými metainformacemi

Tvorba spustitelných archívů

Vytvoříme `jar` s manifestem obsahujícím tento řádek:

`Main-Class:`*`NázevSpouštěnéTřídy`*

poté zadáme:

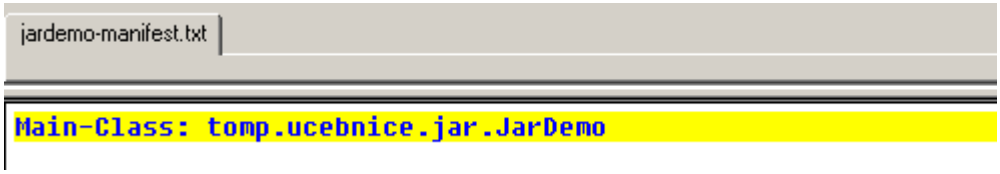
```
java
-jarNázevBalíku
.jar
```

a spustí se metoda `main` třídy ***NázevSpouštěnéTřídy***.

Vytvoření spustitelného archívu - příklad

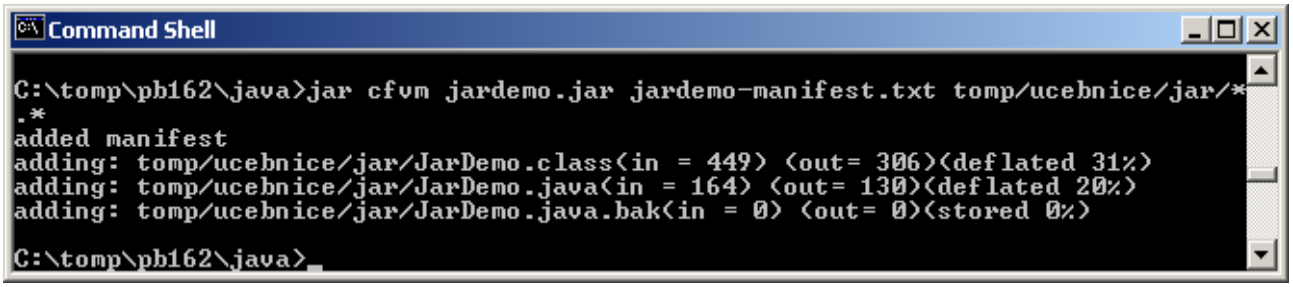
Nejprve vytvoříme soubor manifestu. Příklad jeho obsahu:

Obrázek 11.6. Soubor manifestu



Následně zabalíme archív s manifestem:

Obrázek 11.7. Zabalení archívu s manifestem



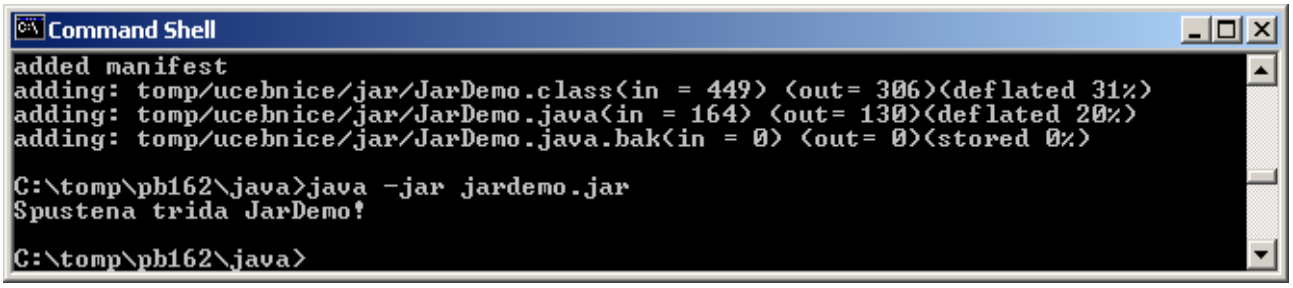
Spuštění archívu - příklad

Spuštění aplikace zabalené ve spustitelném archívu je snadné:

```
java
-jar
 jardemo.jar
```

a spustí se metoda `main` třídy `tomp.ucebnice.jar.JarDemo`:

Obrázek 11.8. Spuštění aplikace z arhcivu



Další příklad spuštění `jar`

- `jar tfv svet.jar | more`
- vypíše po obrazovkách obsah (listing) archívu `svet.jar`.