

Příklad 1

- Napište metodu pro načítání pluginů do programu
 - (neúplná) hlavička metody
 - **`static List loadPlugins(Class pluginInterface, String... pluginNames) ;`**
 - `pluginInterface` – interface, který třídy musejí implementovat
 - `pluginNames` – jména tříd pro natažení
 - upravte hlavičku metody, aby návratový typ mohl být `List`, ze kterého lze přímo (bez přetypování) vybírat instance typu, který reprezentuje třída `pluginInterface`

Příklad 1

- správná hlavička
 - `static <T> List<T> loadPlugins(Class<T> pluginInterface, String... pluginNames);`

Příklad 2

- Napište rozšiřitelný procesor textu
 - program čte ze std. vstupu a vypisuje na std. výstup
 - jako parametr dostane soubor, ve kterém je seznam akcí, které se mají provést s textem
 - akce se provádějí v zadaném pořadí
 - akce ~ třída implementující TextProcessor interface

```
interface TextProcessor {  
    String process(String text);  
}
```
 - soubor obsahuje plná jména tříd implementujících interface

```
File  
cz.cuni.mff.ajava.testprocessor.ToUpperProcessor  
cz.cuni.mff.ajava.testprocessor.JustifyLeft
```

Příklad 3

- Vytvořte rozšiřitelný shell (tj. interaktivní program vykonávající příkazy)
 - help
 - vestavěný příkaz (pouze jeden)
 - vypíše seznam všech dostupných příkazů
 - ostatní příkazy – třídy implementující předepsaný interface
 - navrhnete vhodný interface
 - musí alespoň obsahovat metody pro
 - jméno příkazu
 - nápovědu (pro help příkaz)
 - vykonání příkazu
 - seznam příkazů (tj. tříd) se předá v soubory
 - jako u předchozího příkladu

Příklad 3

- interface pro příkazy např.

```
interface Command {  
    String getCommand() ;  
    String getHelp() ;  
    String execute(String... args) ;  
}
```



Verze prezentace PAJ01.cz.2019.01

Tato prezentace podléhá licenci [Creative Commons Uved'te autora-Neužívejte komerčně 4.0 Mezinárodní License](https://creativecommons.org/licenses/by-nc/4.0/).