

# Platforma Java

---



# Java jako programovací jazyk

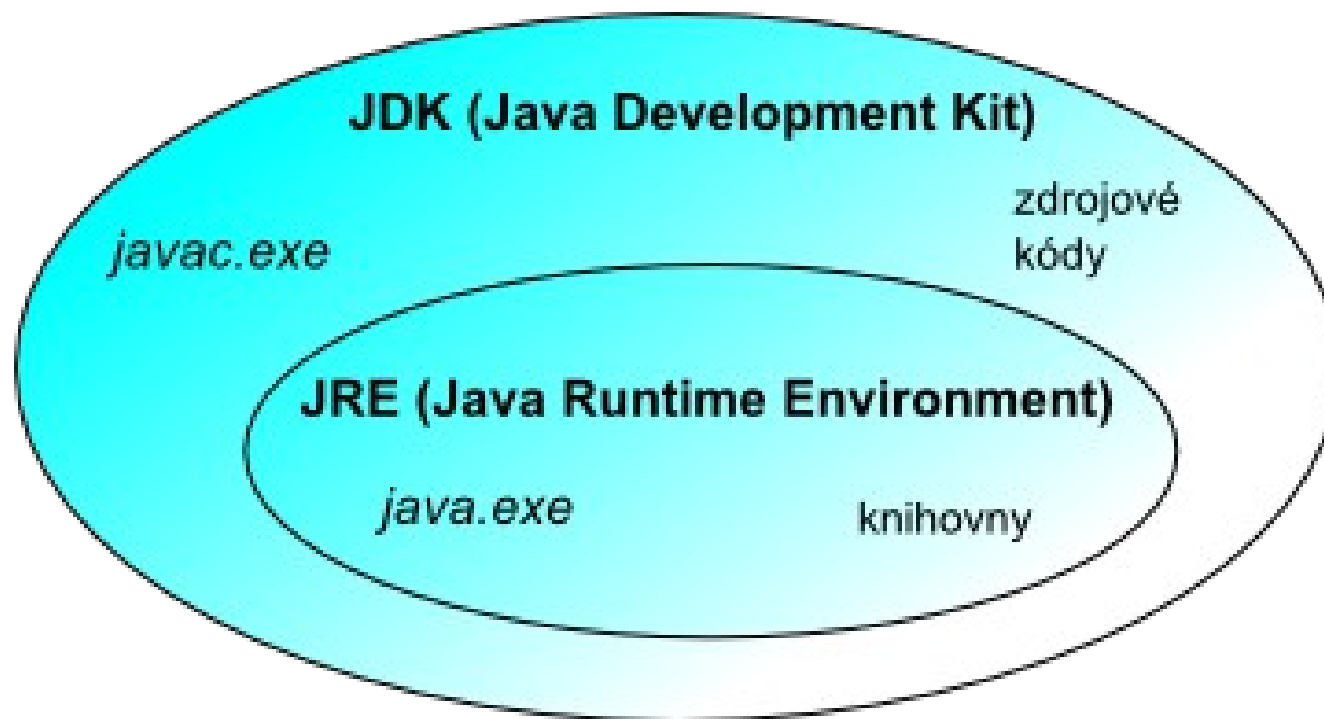
---

- Co je Java
  - Programovací jazyk, vyvinutý firmou Sun Microsystems podle jazyka C++, pro stručný a efektivní zápis programů (1995)
  - Objektivě orientovaný jazyk
  - Nezávislost na platformě
- Co Java není:
  - JavaScript – jazyk vyvinutý firmou Netscape
  - Pouze Applety – ty jsou pouze malou částí Javy
  - J#, Visual J – programovací jazyk od firmy Microsoft pro snadný přechod programátorů, kteří jsou zvyklí programovat v jazyce Java, do prostředí .NET.



# JDK vs. JRE

---



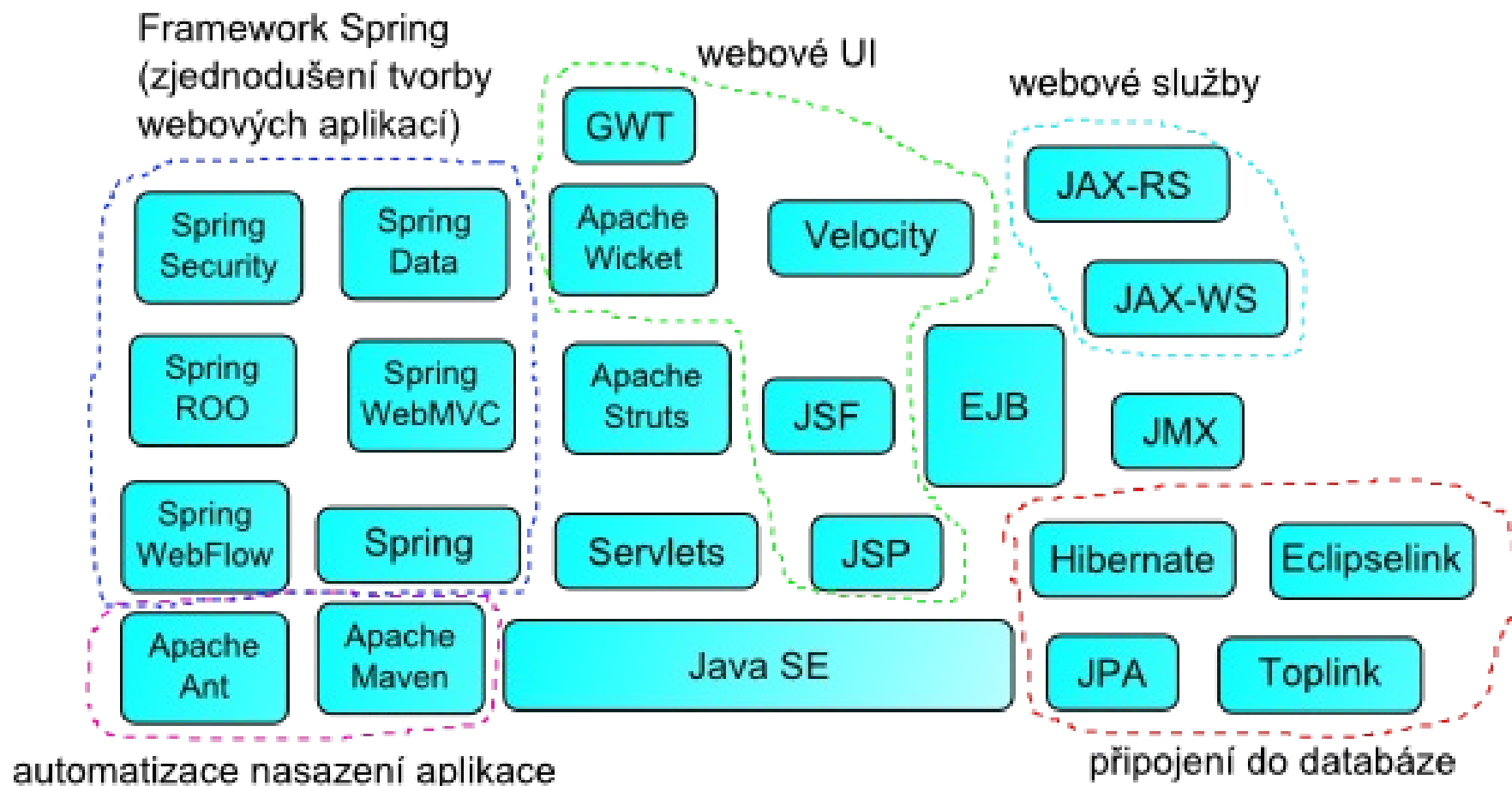


# Java jako platforma

---

- Použití jazyka Java
  - **Java SE** (Java Standard Edition)
    - Desktop aplikace (jako např. C++, C#, Delphi, ...)
    - Applety (malé programy umístěné ve webové stránce)
  - **Java EE** (Java Enterprise Edition)
    - Dynamické webové stránky (jako např. PHP, ASP)
  - **Java ME** (Java Micro Edition)
    - Aplikace pro mobilní telefony
  - **Java FX**
    - „Rich Internet Applications“ pro desktop, mobily, TV – aplikace zaměřené na mobilitu a multimédia
  - Java je dynamický jazyk, neustále se vyvíjí

# Java EE platforma

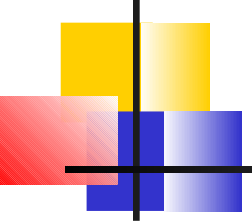




# Vývojová prostředí

---

- Java aplikaci je možné vytvořit i v Notepadu, ale kdo by to dělal, ...
- Pro vývoj se v praxi používají následující vývojová prostředí:
  - **NetBeans IDE** – obsahuje jako první podporu pro nejnovější vlastnosti jazyka Java, ale pokulhává podpora jiných open source technologií, které se při vývoji často používají. Výborná online dokumentace.
  - **Eclipse IDE** – nejpoužívanější vývojové prostředí, které obsahuje celou řadu rozšíření pro efektivní práci s celou řadou Java technologií.
  - **IntelliJ IDEA** – jediné vývojové prostředí, které je placené. Používá se zejména v softwarových společnostech, má celou řadu vychytávek. Má ale i celou řadu nedodělků.
  - **JDeveloper** – vývojové prostředí, které obsahuje velké množství wizardů pro jednoduchou práci s Oracle databází. Jakákoli další práce je ale obyčejně výrazně složitější než při použití jiných prostředí.



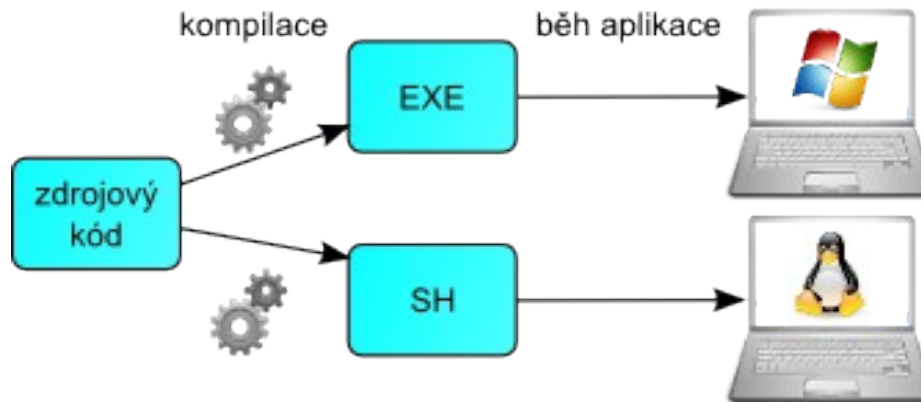
# Java bytecode, JVM (Java Virtual Machine)

---

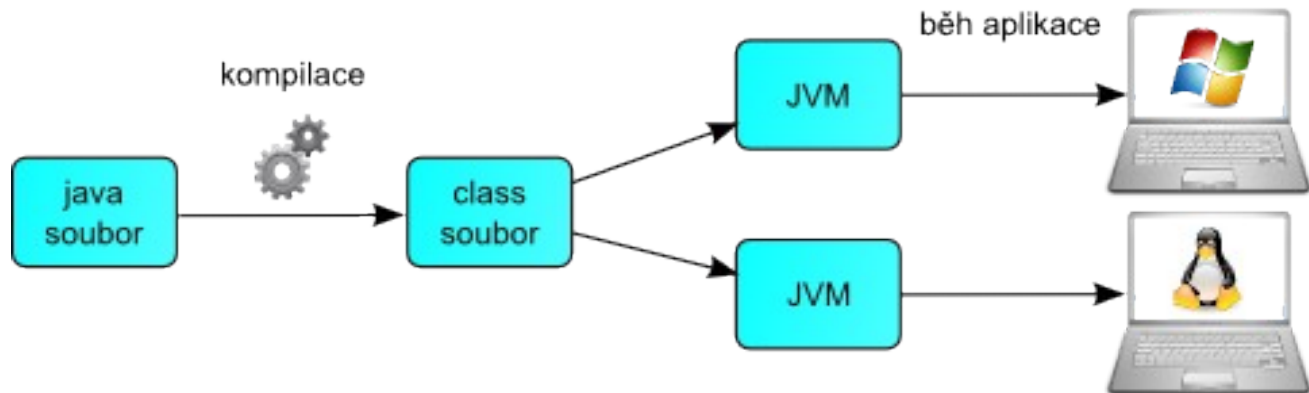
- Nezavislosti na platformě je dosaženo pomocí mezikódu (**bytecode**)
  - Jedná se o jazyk podobný assembleru
  - Při kompilaci se z programu vytvoří posloupnost bytecode instrukcí
  - Při běhu programu interpret (virtual machine) přetransformuje bytecode do strojového kódu a poté se vykoná
  - „Write once, run everywhere“

# Běh aplikace: C/C++/... vs. Java

C/C++/... :



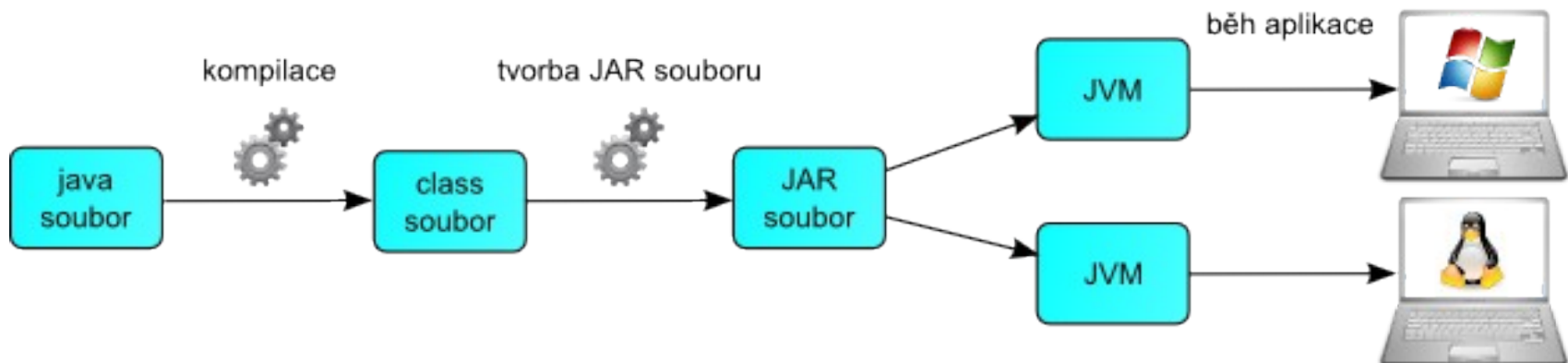
JAVA:





# JAR balíčky

- V typické aplikaci jsou obvykle stovky, tisíce a spíše ještě mnohem více tříd. Z toho důvodu vznikly JAR balíčky. Prakticky se jedná o ZIP archiv, do kterého je možné zabalit class soubory (a obecně jakékoli další soubory) a takový soubor může poté plnit dvě role:
  - Spustitelná Java aplikace (něco jako EXE soubor)
  - Knihovna (něco jako DLL soubor)





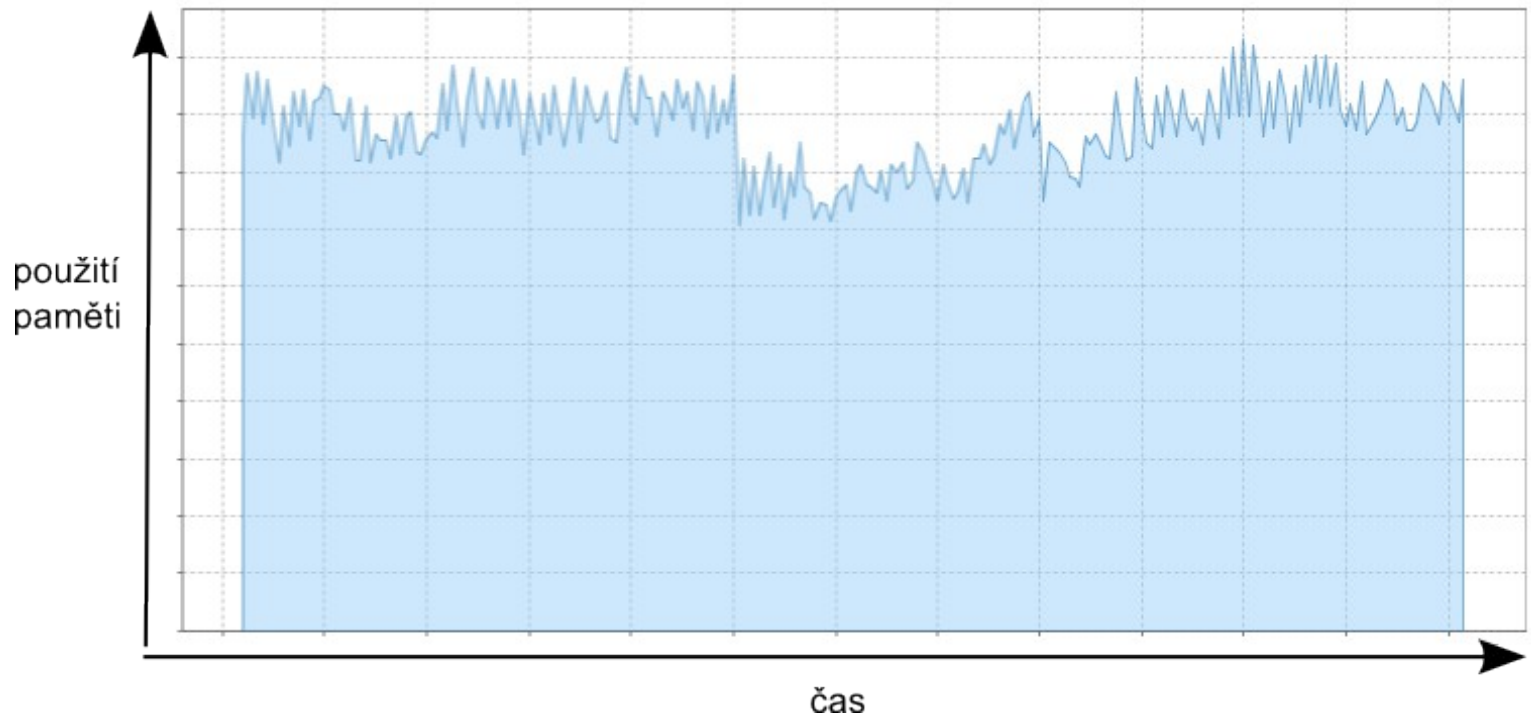
# Spuštění Java aplikace

---

- V případě GUI aplikace stačí poklepat myší na spustitelný JAR soubor. (GUI aplikaci je ale také možné spouštět z příkazové řádky. To má výhodu v tom, že se v konzoli vypisují chybové hlášky v případě, že nepoužíváte nějaký logovací framework).
- V případě konzolové aplikace:
  - Spuštění JAR souboru:
    - `java -jar aplikace.jar`
  - Spuštění class souboru:
    - `java cz.skoleni.aplikace.Main`
    - Pozor na to, že se musíte nacházet v hlavním adresáři aplikace!

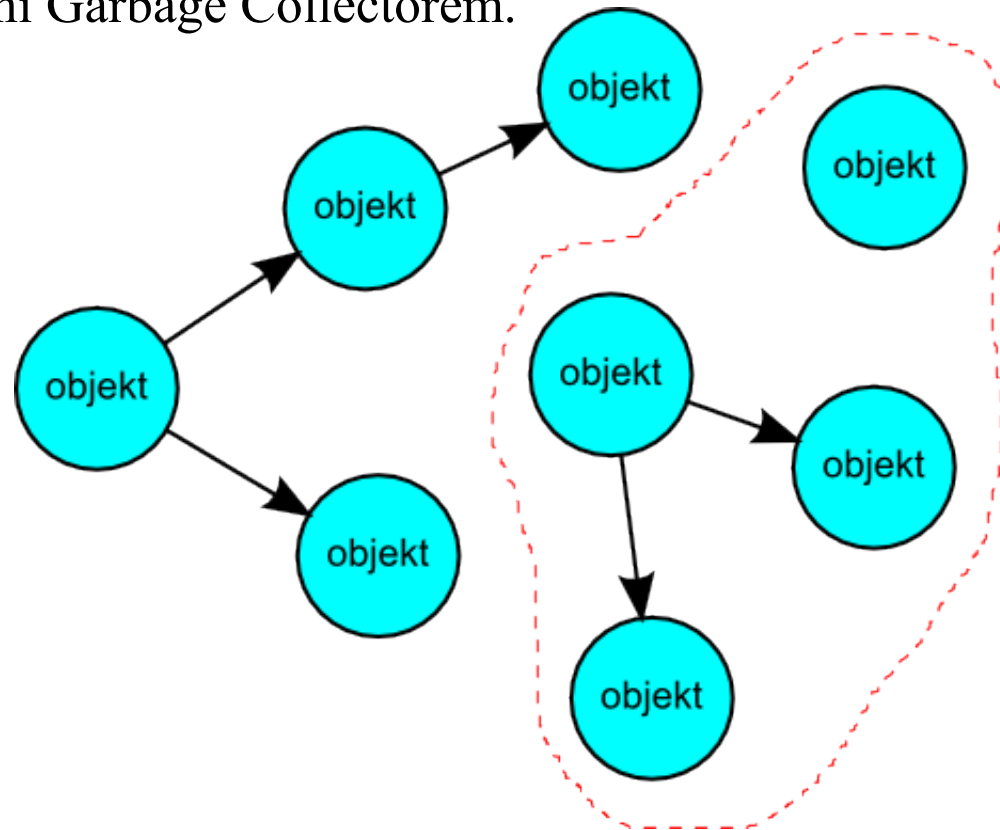
# Garbage Collector I.

- Garbage Collector je mechanismus čištění paměti od neplatných objektů. Díky tomuto mechanismu se programátor nemusí starat o uvolňování paměti (v řadě programovacích jazyků to dělat musí a k tomu existují tzv. destruktory, které v Javě nenaleznete).



# Garbage Collector II.

- V paměti se při běhu aplikace vytváří objektový graf. V případě, že některé objekty zůstanou v tomto grafu osamoceny, pak jsou připraveny k odebrání Garbage Collectorem.





# Garbage Collector III.

---

- Kolik operační paměti aplikace používá?
  - V `C:/Program Files/Java/Jdk/bin` se nachází aplikace `jconsole.exe` a `jvisualvm.exe`. Tyto aplikace se mohou připojit k Java procesu a monitorovat ho. Jedním ze získaných údajů je aktuální použití paměti.
- Kolik operační paměti může aplikace maximálně použít (a jak tuto hodnotu zvýšit)?
  - Standardně Java aplikace může využít pouze 64MB paměti RAM
  - Aby aplikace mohla použít více paměti, musíte při startu aplikace nastavit množství maximální paměti následujícím způsobem:

```
java -jar -Xmx512m aplikace.jar
```

- Nyní bude moci aplikace používat až 512 MB paměti RAM



# Návrh jazyka Java

---

- Platformová nezávislost díky bytecode a JVM
- Ukazatele známé z jazyka C++ byly nahrazeny odkazy (referencemi), které zamezují hrozbě zápisu do neplatné paměti.
- Díky automatické správě paměti (Garbage Collectoru) se programátor nemusí sám starat o uvolňování paměti.
- Serializace pro jednoduché ukládání celých objektů a následné zasílání různými médii.
- Podpora vláken.