



Paralelní a distribuované systémy v Javě

KMI/PDS 2024

Obsah

- Co je to Java + jak fungují vlákna v Javě
- Základy tvorby asynchronního programu (Thread, VirtualThread, ThreadPool, ...)
- Základní synchronizační nástroje (synchronized, ...)
- Další nástroje na synchronizaci (Atomic variables, ...)
- Distribuované systémy v Javě

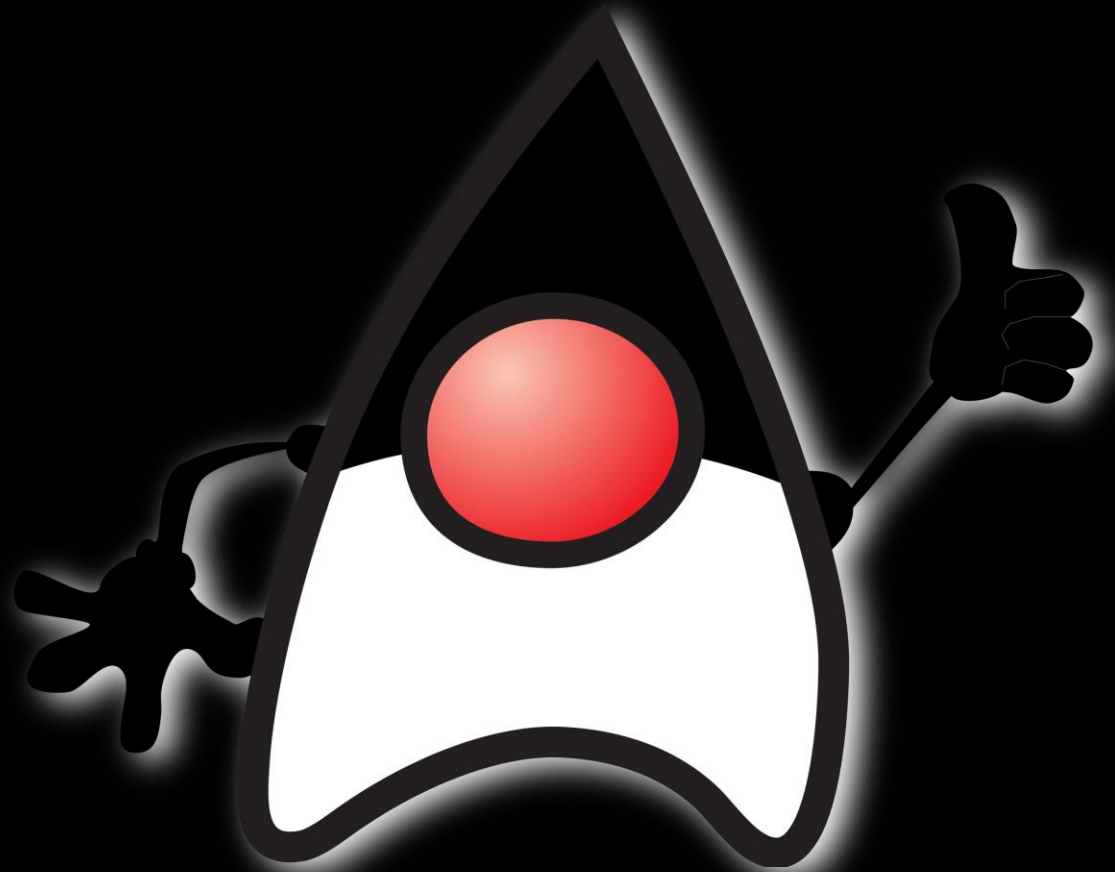
Co je to Java

- Univerzální objektově orientovaný programovací jazyk
- James Gosling (Sun Microsystems) později Oracle
- Co nejméně implementačních závislostí
- "Write Once, Run Anywhere"
- Syntaxe podobná C a C++
- Webové aplikace typu klient server
- V roce 2019 jeden z nejpoužívanějších jazyků (GitHub)



Verze Javy

- každých 6 měsíců nová verze (od roku 2017)
- Nejnovější verze 23 (17. září 2024)
- Long-term Support (LTS): 8, 11, 17, 21



Java Edice

- Java Card
- Java Platform Micro Edition (Java ME)
- Java Platform Standard Edition (Java SE)
- Java Platform Enterprise Edition (Java EE)



PHILIPS

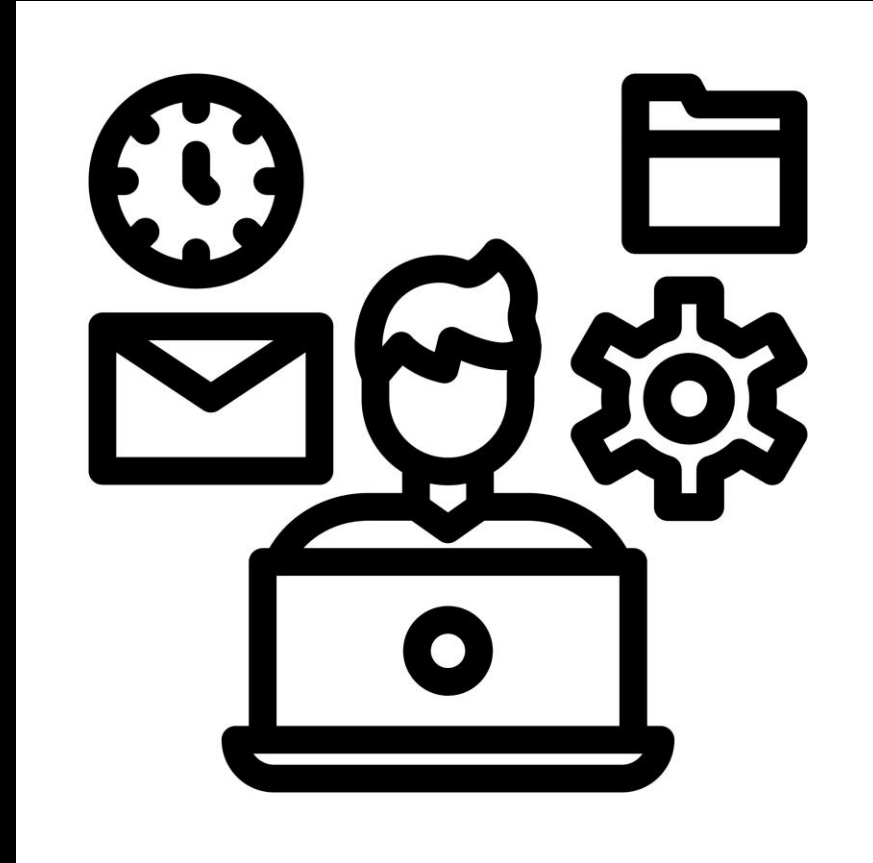
Android SDK

- Jeden z klíčových pilířů
- Pro tvorbu mobilních aplikací
- Nepoužívá žádný z Java standardů (SE, GUI, ME, aj.)
- Bytecode není kompatibilní se standardní Javou
- Dalvik Virtual Machine nebo strojový kód přes Android Runtime



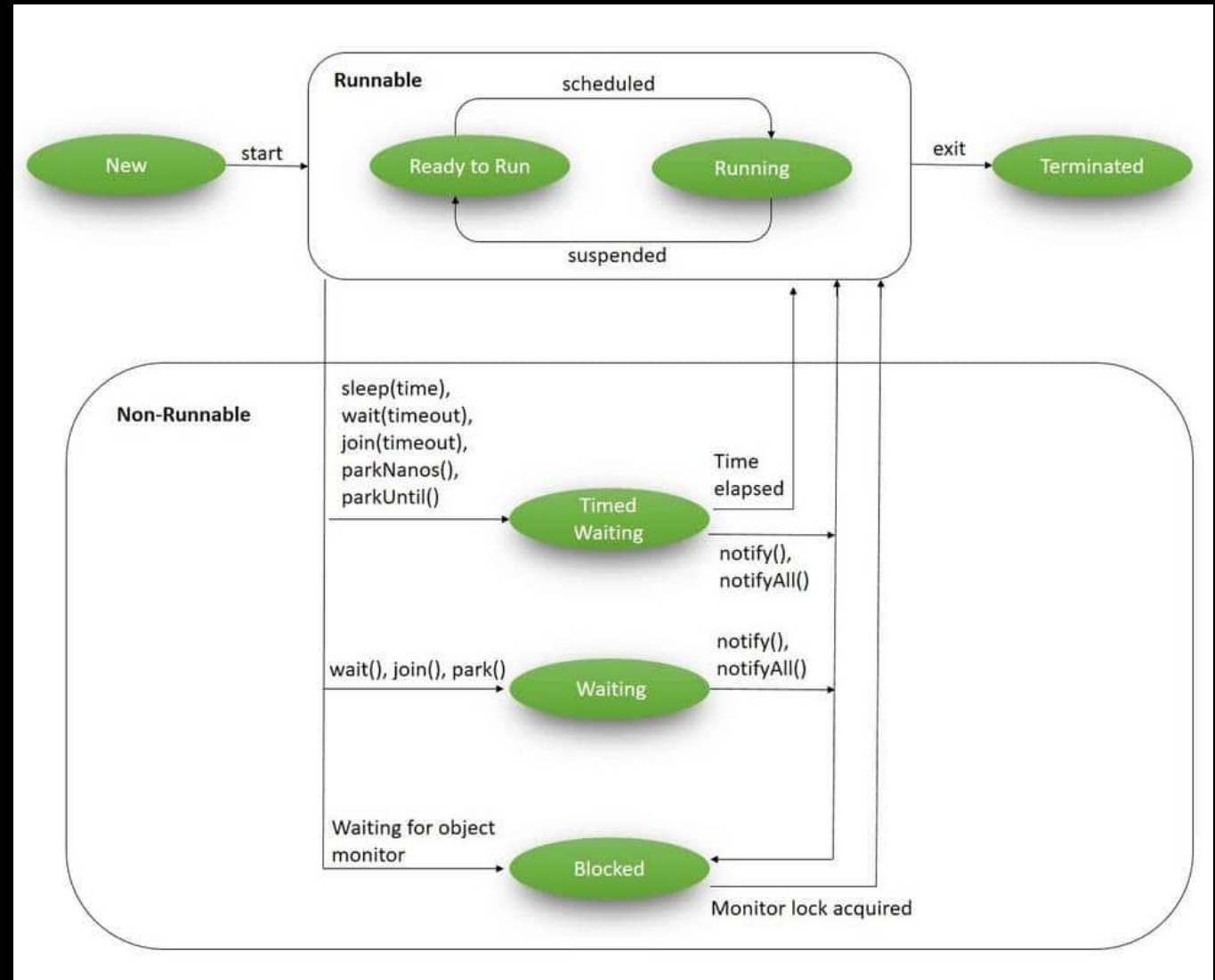
Multitasking

- Více-procesový
 - Není pod kontrolou
- Více-vláknový
 - Je pod kontrolou JVM



Thread Model

- Stavby vlákn:
- New
- Ready to Run
- Running
- Timed Waiting
- Waiting
- Blocked
- Priority
- 1-10



Vytváření vláken

- Thread
 - Vytváří systémové vlákno
 - Na vstupu je třída s rozhraním Runnable nebo lambda výraz
- Virtual Thread
 - Od verze 19
 - Vytváří virtuální vlákno
 - Na vstupu je třída s rozhraním Runnable nebo lambda výraz

Pokročilejší možnosti práce s vlákny

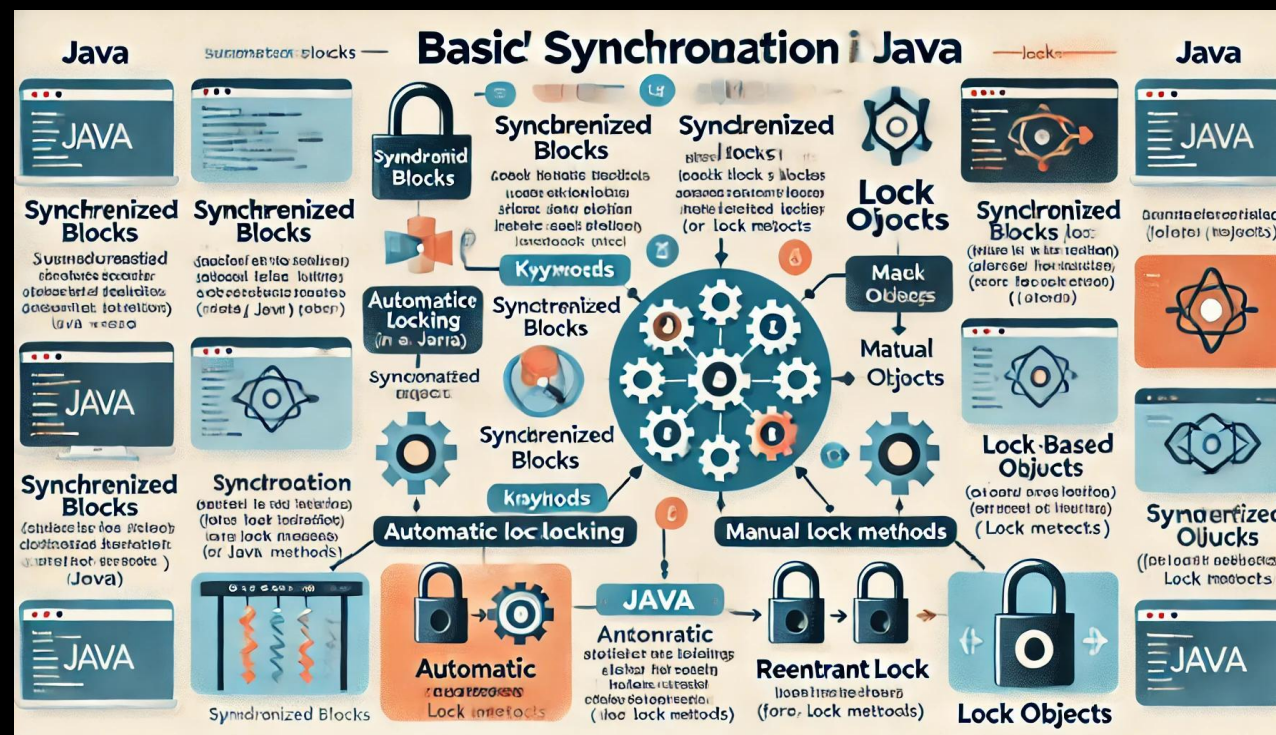
- ThreadPool
 - FixedSize
 - Má maximální počet vláken kterým lze přidělit Task
 - Při spuštění Tasku je mu přiděleno vlákno z pool nebo čeká na uvolnění vlákna.
 - Po dokončení práce je vlákno vráceno do poolu.
 - CachedSize
 - Nová vlákna se vytváří dle potřeby
 - Při spuštění Tasku se buď vytvoří nové vlákno nebo se použije vlákno z poolu
 - Po dokončení práce je vlákno vráceno do poolu.

Pokročilejší možnosti práce s vlákny

- ForkJoinPool
 - Pro práci s úlohami, které lze rekurzivně rozdělit na menší části
 - Využití pro paralelizaci "divide and conquer" úloh
 - Třída RecursiveAction pro úlohy bez návratové hodnoty
 - Třída RecursiveTask<T> pro úlohy s návratovou hodnotou

Základní synchronizační nástroje v jazyce JAVA

- různé úrovně abstrakce
 - "synchronized" vs "lock"



Synchronized

- je klíčové slovo v Javě, které zavádí automatickou synchronizaci nad kritickou sekci kódu
- poprvé představeno v první verzi Javy v roce 1995
- jednoduché použití (k metodě, k bloku kódu v metodě)
- nevýhoda – nižší flexibilita (vlákna co čekají)

```
public synchronized void method()  
{  
    method body  
}
```

```
synchronized (obj) // this is the syntax for a synchronized block  
{  
    Critical section  
}
```

Lock

```
myLock.lock(); // a ReentrantLock object
try
{
    <em>critical section</em>
}
finally
{
    myLock.unlock(); // make sure the lock is unlocked even if an exception is thrown
}
```

```
public void method()
{
    this.intrinsicLock.lock();
    try
    {
        method body
    }
    finally
    {
        this.intrinsicLock.unlock();
    }
}
```

- v knihovně `java.util.concurrent.locks` (od Java 5) – třída pro práci `ReentrantLock`
- manuální kontrola jak zamykání probíhá
- při práci s lock používáme try-finally bloky (odemknutí při vyskytnutí výjimky)
- použití
 - manuální uzamčení a odemknutí
 - časově omezené čekání (voláme `tryLock(čas, jednotka času)` – maximální doba na kterou bude vlákno čekat)
 - spravedlivý přístup (`ReentrantLock` s parametrem `true`)

Vsuvka – Stav objektů (**await()**, **signal()**, **signalAll()**)

- můžeme chtít, aby vlákno čekalo dokud není splněná nějaká podmínka (Condition)
 - **await()** – vlákno, které zavolá bude čekat (bude blokováno)
 - **signal()** – probudí jedno vlákno, které čeká (obvykle vlákno, které čeká nejdéle, ale není to garantováno)
 - **signalAll()** – probudí všechna vlákna, které čekají
- platí i u synchronized metod a bloků (**wait()**, **notify()**, **notifyAll()**)
- rozdíly:
 - synchronized - pracuje se zámkem na úrovni objektu (monitoru), ...
 - condition u lock - odděluje zamykání a podmíněné proměnné, což umožňuje větší flexibilitu, ...

Semaphore

- v knihovně `java.util.concurrent` (od Java 5) – třída pro práci Semaphore
- efektivní způsob řízení přístupu k omezenému počtu zdrojů pomocí počítadla dostupných míst (klasický semafor)
- operace **acquire()** (čeká dokud nezíská místo) a **release()** (vrací místo a umožňuje vstoupit dalším vláknům)
- 2 druhy v jazyce java
 - neférový (`new Semaphore(2)`)
 - férový (`new Semaphore(2, true)`) (férový – vlákna získají přístup v pořadí, v jakém o něj žádala)

Monitor

- představil Per Brinch Hansen v 70. letech 20. Století
- metody synchronized už jsou monitory
- vlastnosti v jazyce Java
 - musí to být třída s private atributama
 - každá metoda je synchronized (můžou ta být i podmínky wait(), notify(), notifyAll())

Další synchronizační nástroje

- **CountDownLatch** a **CyclicBarrier** jsou pro synchronizaci skupin vláken
- **Exchanger** umožňuje výměnu dat mezi dvěma vlákny
- **ReadWriteLock** a **StampedLock** jsou pokročilé zámky pro optimalizaci čtení a zápisu
- **Phaser** je flexibilní synchronizační nástroj pro řízení fází

Atomické proměnné

- Obaly pro vybrané datové typy, poskytující atomické operace
 - AtomicInteger, AtomicIntegerArray, AtomicIntegerFieldUpdater
 - AtomicLong, AtomicLongArray, AtomicLongFieldUpdater
 - AtomicBoolean, AtomicReference,...
- Většina poskytuje metody `get()` a `set()`
 - Atomicky získá (nastaví) hodnotu
- Dále také další metody dle typu, např.
 - `getAndSet(newValue)` – Atomicky nastaví na novou hodnotu a vrátí starou
 - `getAndAdd(delta)` – Atomicky přičte hodnotu a vrátí starou
 - `compareAndSet(expectedValue, newValue)` – Atomicky změní hodnotu, pokud má očekávanou hodnotu

AtomicInteger

- Zapouzdřuje a poskytuje atomické operace pro hodnotu typu int
- `getAndIncrement()`
 - Atomicky inkrementuje hodnotu a vrátí původní hodnotu
- `getAndAdd(int delta)`
 - Atomicky přičte hodnotu a vrátí původní hodnotu
- `compareAndExchange(int expectedValue, int newValue)`
 - Atomicky nastaví na novou hodnotu, pokud má očekávanou hodnotu. Vrátí původní hodnotu
- `compareAndSet(int expectedValue, int newValue)`
 - Atomicky nastaví na novou hodnotu, pokud má očekávanou hodnotu. Vrátí, zda došlo k nastavení hodnoty

AtomicIntegerFieldUpdater

- Umožňuje použití atomických operací na daném `volatile int` atributu dané třídy
- Vytvoření pomocí `AtomicIntegerFieldUpdater.newUpdater(Clazz.class, "field")`
- Používá reflexi
- Méně bezpečné než `AtomicInteger` (v případě `compareAndSet`)

Atomické proměnné

- Dále také:
 - AtomicIntegerArray
 - Stejné metody jako AtomicInteger, pouze mají další argument pro index prvku
 - LongAdder
 - Pro zpracování velkého množství atomických inkrementací
 - Uchovává si množinu buněk, které jednotlivá vlákna upravují pomocí add() a increment()
 - Pro získání celkové hodnoty všech buněk slouží sum()
 - LongAccumulator
 - Zobecnění LongAdder pro binární funkce na long
 - Funkce musí být komutativní a asociativní

Souběžné kolekce

- Běžné kolekce nepodporují manipulaci více vlákeny - chování není garantováno
 - Vyvolání `java.util.ConcurrentModificationException`
 - Vyvolání jiné výjimky
 - Nekonzistentní stav - chybějící prvky
- Existují thread-safe verze běžných kolekcí:
 - `ArrayList` - `CopyOnWriteArrayList`
 - `HashMap` - `ConcurrentHashMap`
 - `ArraySet` - `CopyOnWriteArraySet`
 - `LinkedList` - `LinkedBlockingDeque`
- Bezpečné přidávání, odebírání, vyhledávání prvků, použití iterátorů a proudů

Synchronizované kolekce

- Sada statických metod k vytvoření thread-safe kolekcí z běžných kolekcí
- `Collections.synchronizedList`, `...synchronizedSet`, `...synchronizedMap`, ...
- Vytvoří objekt obalující původní kolekci, implementující stejné rozhraní
 - V případě `Collections.synchronizedList` jakákoliv instance `List<T>`
- Je třeba přistupovat přes novou kolekci
- Všechny její metody používají společný mutex
 - Metody nové kolekce volají metody původní kolekce ze `synchronized` bloku
 - Kteroukoliv metodu může používat v jednu chvíli pouze jedno vlákno
- Při použití iterátorů a proudů je třeba ručně zamknout pomocí `synchronized`

Paralelní proudy

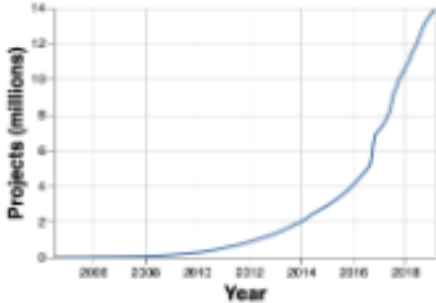
- Nově vytvořené proudy (pomocí `Collection.stream`, `Stream.of`) jsou sekvenční
- Ze sekvenčního proudu lze vytvořit paralelní - `Stream.parallel`
- Případně z kolekce pomocí `Collection.parallelStream` (výsledek nemusí být paralelní)
- Rozdělení původního proudu na více podproudů, které jsou zpracovávány více vlákny
- Používá `Splitterator` a `ForkJoinPool` (`ForkJoinPool.commonPool`)
- Vhodné pro velké kolekce, nezávislé operace, výpočetně náročné operace
- Nevhodné pro malé kolekce, I/O-Bound operace, závislé operace

Možnosti DS v Javě

MVN REPOSITORY

Search for groups, artifacts, categories

Indexed Artifacts (45.6M)




Year	Projects (millions)
2008	0.1
2010	0.5
2012	1.0
2014	2.0
2016	5.0
2018	13.0

Popular Categories

Testing Frameworks & Tools

Home » Repositories » Central

**Central Repository**
<https://repo1.maven.org/maven2/>

URL	https://repo1.maven.org/maven2/
Storage	49359.0 GBs
Packages	15,235,882 indexed packages

Základní knihovny

- **Remote Method Invocation (RMI)**

- Umožňuje objektům volat metody na vzdálených objektech (Remote Procedure Call - RPC)
- `Package java.rmi`

- **Java Message Service (JMS) - Jakarta Messaging**

- Standard pro messaging, který umožňuje komponentám aplikace (Java EE) vytvářet, odesílat, přijímat a číst zprávy.

- **Open Message Queue (OpenMQ)**

- Implementace JMS
- Podpora Point-to-Point a Publish/Subscribe modelů

- **RESTful Služby**

- Framework JAX-RS (implementace: Jersey, RESTEasy) pro vytváření RESTful API.

- **SOAP Služby**

- Použití JAX-WS pro SOAP-based webové služby.

Nadstavba frameworků

- **Spring Boot**

- Rozsáhlý framework
 - Mikroservisní architektura - Modularita
 - REST, SOAP, WebSocket (STOMP)
 - + Spring Cloud = Service Discovery & Load Balancing...
 - + Kubernetes = Volba lídra
 - + Resilience4j = Circuit Breaker

- **JGroups**

- Nástroj pro komunikaci v DS
 - Clustering aplikací - synchronizovat stav aplikace mezi více instancemi
 - Volba lídra

Hazelcast

- Distribuované Kešování / Distribuovaná Paměť
- Sdílené Datové Úložiště
 - Atomické operace
- Distribuované Výpočty a Paralelní Zpracování
 - MapReduce (model pro zpracování velkých množin dat pomocí paralelního zpracování)

