01. Programovací jazyk JAVA (historie, zařazení, průběh překladu)

Historie

Green Team

Na počátku zrodu Javy byl vznik Green Teamu vedený Jamesem Goslingem, který měl za úkol vytvořit pro společnost Sun Microsystems jazyk, který bude stručný a efektivní. Pro programování systémů domácích spotřebičů (embedded systémy).

Oak

Green Team se zprvu pokoušel systém pro spotřebiče založit na jazyce C++, poté i na jazyku Pascal. Po neúspěších Green Team navrhl nový jazyk, jenž pochází z C++. Jazyk pojmenovali dle stromu, co rostl před kanceláří Jamese Goslinga, Oak (Dub). Oak se zprvu nemohl prosadit, po neúspěších v PDA telefonech a set-top boxů k televizorům použit pro webové aplikace – applety. Oak oživil v té době statický web a začal se proto používat na psaní webových appletů (1991).

Java

V roce 1995 se zjistilo, že jazyk se jménem Oak už existuje a tak dostal název Java. Java byla poprvé představena v roce 1995 na konference SunWord. Technologie Java byla licencována společnostem IBM, Microsoft, Symantec, Silicon Graphics, Inprise (Borland), Oracle, Toshiba či Novell. V roce 1996 bylo vydáno první Java Develepement Kit – JDK 1.0, které obsahovalo vše potřebné pro tvorbu appletů.

C++

Java pochází z jazyka C++, byly odstraněny konstrukce, které se nepoužívaly, způsobovaly těžké běhové chyby a pointery. Někteří programátoři co přešli z jazyka C++ k Javě tvrdí, že v Javě jejich produkce programů dvojnásobně stoupla.

JVM

Java běží na Java Virtual Machine, která zprostředkovává veškeré propojení s hardwarem, což zaručuje nezávislost na OS a hardwaru počítače. Ovšem i to má svou nevýhodu, operační systémy musí mít JVM nainstalovanou. Na konci 90. let se totiž Sun dostal do sporu s Microsoftem, který si chtěl Javu upravovat podle sebe. Vše to vyvrcholilo soudním sporem a nakonec odebráním licence Microsoftu.

Dnes

Dnes je Java jeden z nejpoužívanějších programovacích jazyků. Programy v Javě se píší pro mobily, osobní počítače, servery a internetové aplikace. Největší zastoupení má Java na serverových aplikacích.

Překlad a spuštění

JVM

Java Virtual Machine obstarává vazbu na hardware a na JVM běží všechny java programy. JVM je součástí balíčků jak JDK (Java Develepement Kit) tak i JRE (Java Runtime Environment).

JDK

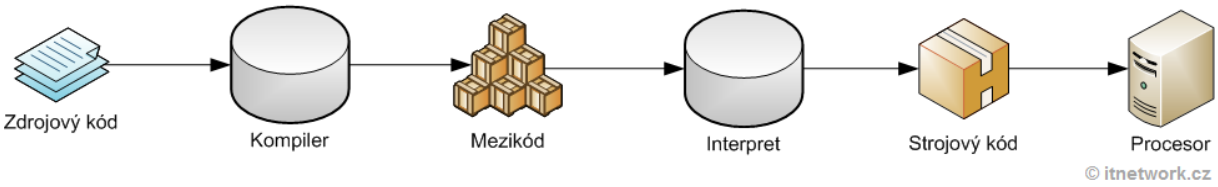
Java Develepement Kit obsahuje JVM a také spoustu předepsaných tříd uspořádaných v balíčcích v jedné velké knihovně. JDK je potřeba pro překlad programu.

JRE

Java Runtime Environment obsahuje JVM a knihovny stejně jako JDK, hlavní rozdíl je, že JRE je určeno jen pro spouštění javovských aplikací, nikoliv pro překlad.

Překlad

Zdrojový kód je nejprve přeložen (**kompilerem**) do tzv. mezikódu (**bajtkód**). Jedná se v podstatě o binární kód, který má jednodušší instrukční sadu a přímo podporuje objektové programování. Tento mezikód je potom díky jednoduchosti relativně rychle interpretovatelný virtuálním strojem (**interpretem - JVM**). Výsledkem je strojový kód pro procesor.



JIT Compiler

V době zavedení programu z disku do paměti počítače (Po ověření správnosti bajtkódu) jej přeloží do strojového jazyka konkrétního počítače (prakticky vytvoří v paměti EXE soubor). Ten pak běží stejnou rychlostí jako kterýkoliv jiný zkompilovaný program napsaný třeba v C.

Paměť:

Paměť javy je rozdělena do 4 segmentů:

Data Segment

Obsahuje globální a statické údaje (data), které jsou přímo inicializována uživatelem. Další část Data Segmentu je BSS (Block Start with Symbol), kde OS inicializuje paměťové bloky na 0, tato oblast je fixována a má statickou velikost.

Code Segment

Segment, kde se nachází aktuální zkompilovaný Java code (bajtkód)

Stack Segment

Obsahuje veškeré vytvořené objekty v běhu.

Heap

Zde se nacházejí deklarované proměnné.

Výhody:

- **Odhalení chyb ve zdrojovém kódu** – Díky kompilaci do bytekódu se jednoduše odhalí chyby ve zdrojovém kódu.
- **Stabilita** – Díky tomu, že interpret kódu rozumí, zastaví před vykonáním nebezpečné operace a na chybu upozorní
- **Jednoduchý vývoj** – Jsou k dispozici hitech datové struktury a knihovny, správu paměti za nás provádí garbage collector.
- **Rychlost** – Rychlost se u virtuálního stroje pohybuje mezi interpretem a kompilerem. Virtuální stroj již výsledky své práce po použití nezahazuje, ale dokáže je **cachovat**, sám se tedy optimalizuje při četnějších výpočtech a může dosahovat až rychlosti kompileru. Start programu bývá pomalejší, protože stroj překládá společné využívané knihovny.
- **Málo zranitelný kód** – Aplikace se šíří jako zdrojový kód v bytekódu, není tedy úplně jednoduše lidsky čitelná.
- **Přenositelnost** – Hotový program poběží na každém HW, kde je virtuální stroj.

Další výhodou Javy je, že je zcela zdarma a tedy dostupná všem vývojářům. Aplikace v Javě lze také spouštět přímo ve webovém prohlížeči pomocí Java Web Start.

Jazyky s virtuálním strojem ctí objektově orientované programování a jedná se o současný vrchol vývoje v této oblasti.

Zařazení

Jednoduchý, Objektové orientovaný, distribuovaný, interpretovaný, bezpečný, nezávislý, přenositelný, procedurální

- **Jednoduchý** – Jeho syntaxe je zjednodušenou (a drobně upravenou) verzí syntaxe jazyka **C** a **C++**. Odpadla většina konstrukcí, které způsobovaly programátorům problémy, na druhou stranu přibyla řada užitečných rozšíření.
- **Objektově orientovaný** – s výjimkou osmi primitivních datových typů jsou všechny ostatní datové typy **objektové.**
- **Distribuovaný** – Je navržen pro podporu aplikací v síti (podporuje různé úrovně síťového spojení, práce se vzdálenými soubory, umožňuje vytvářet distribuované klientské aplikace a servery.
- **Interpretovaný** – místo skutečného strojového kódu se vytváří pouze tzv. **mezikód** (bytekód). Tento formát je nezávislý na **architektuře počítače** nebo zařízení. Program pak může pracovat na libovolném počítači nebo zařízení, které má k dispozici interpret Javy (JVM). V současných verzích Javy se využívá **JIT compiler**. Tato vlastnost zásadním způsobem zrychlila provádění programů v Javě, ale výrazně zpomalila start programů. V současnosti se převážně používají technologie zvané HotSpot compiler, které mezikód zpočátku interpretují a na základě statistik získaných z této interpretace později provedou překlad často používaných částí do strojového kódu včetně dalších dynamických optimalizací (jako je např. inlining krátkých metod atp.).
- **Robustní** – Je určen pro psaní vysoce spolehlivého **softwaru** – z tohoto důvodu neumožňuje některé programátorské konstrukce, které bývají častou příčinou chyb (např. správa paměti, příkaz goto, používání ukazatelů). Používá tzv. silnou **typovou kontrolu** – veškeré používané proměnné musí mít definovaný svůj datový typ.

- **Generační správa paměti** – Správa paměti je realizována pomocí automatického **Garbage collectoru,** který automaticky vyhledává již nepoužívané části paměti a uvolňuje je pro další použití. To bylo v prvních verzích opět příčinou pomalejšího běhu programů. V posledních verzích běhových prostředí je díky novým algoritmům pro garbage collection a tzv. generační správě paměti (paměť je rozdělena na více částí, v každé se používá jiný algoritmus pro garbage collection a objekty jsou mezi těmito částmi přesunovány podle délky svého života) tento problém ze značné části eliminován.
- **Bezpečný** – Vlastnosti, které chrání počítač v síťovém prostředí, na kterém je program zpracováván, před nebezpečnými operacemi nebo napadením vlastního operačního systému nepřátelským kódem.
- **Nezávislý na architektuře** – Vytvořená aplikace běží na libovolném operačním systému nebo libovolné architektuře. Ke spuštění programu je potřeba pouze to, aby byl na dané platformě instalován správný virtuální stroj. Podle konkrétní platformy se může přizpůsobit vzhled a chování aplikace.
- **Přenositelný** – Vedle zmíněné nezávislosti na architektuře je jazyk nezávislý i co se týká vlastnosti základních datových typů (je například explicitně určena vlastnost a velikost každého z primitivních datových typů). Přenositelností se však myslí pouze přenášení v rámci jedné platformy Javy (např. J2SE). Při přenášení mezi platformami Javy je třeba dát pozor na to, že platforma určená pro jednodušší zařízení nemusí podporovat všechny funkce dostupné na platformě pro složitější zařízení a kromě toho může definovat některé vlastní třídy doplňující nějakou speciální funkčnost nebo nahrazující třídy vyšší platformy, které jsou pro nižší platformu příliš komplikované.
- **Výkonný** – Přestože se jedná o jazyk interpretovaný, není ztráta výkonu významná, neboť překladače pracují v režimu **JIT** a do strojového kódu se překládá jen ten kód, který je opravdu zapotřebí.
- **Více úlohový** – Podporuje zpracování více vláknových aplikací
- **Dynamický** – Java byla navržena pro nasazení ve vyvíjejícím se prostředí. Knihovna může být dynamicky za chodu rozšiřována o nové třídy a funkce, a to jak z externích zdrojů, tak vlastním programem.
- **Elegantní** – Velice pěkně se v něm pracuje, je snadno čitelný (např. i pro publikaci algoritmů), přímo vyžaduje ošetření výjimek a typovou kontrolu.