Java a Tvorba Softwaru '98

Martin Fussek, Katedra měřicí a řídicí techniky, VŠB-Technická univerzita Ostrava, FEI, tř. 17. listopadu, 708 33 Ostrava – Poruba, Česká republika
E-mail: Martin.Fussek@vsb.cz, http://kat455.vsb.cz

Abstrakt

V současné době je *Java světovým programovacím jazykem* číslo jedna. Příspěvek se snaží nastínit proč je Java mezi tvůrci software tak oblíbená se zřetelem na praktickou stránku vztahu *„Java a Tvorba softwaru"*. Obsahuje stručnou charakteristiku jazyka Java, její objektovou povahu a podporu pro tvorbu softwaru z komponent. V neposlední řadě nastiňuje možností jejího použití v informačních a řídicích systémech, Internetu a Intranetu. Příspěvek se také zmiňuje o JavaScripu – který v poslední době nabývá na významu ve vztahu k dynamic HTML a jež je jazyku Java je velmi blízký.

Příspěvek vznikl na základě práce na projektu grantové agentury české republiky "Vývojové a softwarové nástroje pro přenositelné a bezpečné aplikace v řízení průmyslových procesů" (číslo 102/97/0542).

1. Úvod

Java je skutečný světový fenomén. Java je jedním z několika málo velkých úspěchů devadesátých let: nový programovací jazyk, který svým vlastním pohledem na svět vytvořil světovou bombu. Sun uvolnil alfa verzi jazyka Java na začátku roku 1995 a ačkoliv byla stěží známa a velmi málo používaná ještě začátkem roku 1996 je Java v současnosti jedním z nejmódnějších slov. Ovšem nebyl to samotný jazyk co způsobilo tento neuvěřitelný boom. Již v minulosti zapadly v zapomnění jazyky jiné, možná lepší. To co všechny přivábilo k jazyku Java byl pojem *applet*, *mini aplikace* která pracuje uvnitř Web stránky.

Žádná z vlastností jazyka Java není technicky nová. Bezpečné jazyky existovaly dávno před ní, včetně výkonově kompatibilních (Reiser a Wirth, 1992). Virtuální stroje s instrukcemi bytového kódu (další vítězný aspekt *Java zajišťující platformovou* nezávislost) byly známy již v dobách dávno minulých, vzpomeňme Pascal p-machine (Nori a další, 1981) a Smalltalk 80 virtual machine (Krasner, 1983). Dokonce i koncepce objektové integrace na Web byla demonstrována již dříve. Základním úspěchem jazyka ovšem Java bylo (a je) že sloučil všechno dobré a vypustil výsledek ve správnou dobu.

2. Důvody předcházející vzniku jazyka Java

V současné době můžeme bez nadsázky říci, že World Wide Web „bere svět útokem". Web začíná být velmi všední a všeobecně známý a jeho aktivní využívání začíná být pro většinu (nejen) mladé generace stejně prosté jako používání telefonu. Web se každou minutou rozrůstá a Web stránky vznikají závratnou rychlostí a každá se snaží oznamovat celému světu „Já jsem ta nejlepší, podívej se na mne!". Počet stránek vyvolává bouřlivou soutěž o návštěvníky klade velmi vysoké nároky na návrháře stránek, kteří se musí snažit vytvářet stránky stále více a více přitažlivější. Bohužel možnosti co lze vytvořit se statickým HTLM (Hypertext Markup Language) jsou omezené. A jako obyčejně to začali být programátoři, začali hledat způsob jak navrhovat a vytvářet stránky interaktivní, dynamické, obecně řečeno „živé".

Na první pohled to nevypadá jako velký problém. Někdo napíše svůj program ve svém oblíbeném programovacím jazyce (jako je C, C++, Delphi, Visual Basic, Component Pascal), zkompiluje jej a uloží výsledný spustitelný kód na server společně s HTML stránkami. Asi by bylo nutné menší rozšíření HTML jazyka tak, aby umožnil Web prohlížeči automaticky natáhnout požadovaný program ze serveru a spustit jej na klientově počítači. Tento program by mohl otevřít svá vlastní okna, dialogové boxy, a teoreticky dělat cokoliv (třeba i přehrávat audio/video) *Voila* dosáhli jsme požadovaného - vytvořili jsme živé Web stránky.

Naneštěstí tento jednoduchý postup má několik závažných nedostatků. První, konvenční programovací jazyky nejsou navrženy pro spouštění na Web. Nemají vlastnosti potřebné pro vykonání dokonce ani nejjednodušších obecných „Web" operací. Programátoři by si proto museli napsat své vlastní funkce na provedení i těch nejzákladnějších operací, jako je například natahování souborů s obrázky ze serveru, tak aby mohly být zobrazeny v programu na klientově stroji. Programování pro Web v konvenčních, ne na Web orientovaných jazycích, by se velmi rychle stalo dost problematické a náročné. Navíc vynalézavost programátorů by nevyhnutelně směřovala k množství různých řešení stejného problému, což by značně komplikovalo život jiným programátorům, kteří by se nažili číst a pochopit každý jednotlivý program (či řešení).

Kromě toho, konvenční programy generují dosti velké spustitelné soubory. Pochopitelně, že „dosti velké" je relativní pojem. 500 kB je téměř nic pokud budeme spouštět program přímo z našeho harddisku. Takový program bude z disku zavedený a připravený k použití (obrazně řečeno) ještě předtím, než uvolníme prst z tlačítka myši. Nicméně s modemem s přenosovou rychlostí 28,8 kilo-bitů za sekundu (Kbps) bude natažení úplně stejného programu skrz telefonní linku trvat více než dvě minuty, a to nepředpokládáme žádné komunikační prodlevy způsobené přeplněnými linkami.

Třetím významným aspektem je také otázka bezpečnosti. Skutečně bychom chtěli aby náš prohlížeč spustil nějaký program ze sítě bez možnosti naší kontroly? Jakmile by byl daný program natažen na klientův (tedy náš) počítač a byl by spuštěn, mohl by si začít dělat co by chtěl. Napíši-li si program na vymazání svého harddisku a spustím jej (proč asi?) program ochotně vymaže můj harddisk a já mohu

obviňovat pouze sám sebe. (Nebo bych mohl zažalovat výrobce kompilátoru?). Na druhé straně, pokud narazím na něčí WWW stránku a tato stránka mi natáhne nějaký program na můj počítač a ten zlikviduje obsah mého disku, myslím že mohu být spravedlivě naštvaný a asi bych si takový program na svůj počítač nenatahoval, kdybych předem věděl, že může zlikvidovat obsah mého disku.

Neméně významný je problém různých výpočetních platforem. Na Web je připojeno velké množství různých typů počítačů a to vyvolává základní otázku „Jakou platformu by měly naše programy podporovat?". Dejme tomu, že většina domácích počítačů připojených k síti je (nepochybně) typu PC, ale je zde také mnoho počítačů typu Macintosh, UNIX a podobně. Jistěže můžeme vymyslet schéma podle kterého by prohlížeč mohl hostitelský počítač identifikovat serveru - pokud je počítač typu PC, pošli tento spustitelný soubor, pokud je typu Macintosh, pošli zase tento. Ale ani to není dostačující. Ve světě PC mohou lidé používat různé operační systémy Microsoft Windows 3.1, Microsoft Windows 95, Microsoft Windows NT a OS/2. Na tomto místě vzpomeňme specificky český problém vyvolaný existencí nejméně pěti různých kódování češtiny, které komplikují život všem českým tvůrcům Web stránek (a programátorům obecně) a dosud nebyl spolehlivě vyřešen.

3. Java

Java je: jednoduchý, objektově orientovaný, distribuovaný, interpretovaný, robustní, bezpečný, nezávislý na architektuře, přenosný, vysoce výkonný, víceprocesní a dynamický jazyk.

Jazyk Java byl tedy navržen tak aby řešil (?!) všechny v předchozí kapitole zmíněné problémy stejně tak jako množství dalších.

Java obsahuje zabudovanou *podporu pro Web* (a síť obecně, je tedy *distribuovaný* jazyk), *čímž zbavuje programátory nepříjemného břemene řešení obecných Web problémů.*

Java je *hardwarově nezávislá*. Stejný program může být vykonán na počítači PC, Macintosh, nebo dokonce na UNIX (pokud je zde nainstalován prohlížeč který podporuje jazyk Java). S nezávislostí na platformě úzce souvisí fakt, že Java je jazyk *interpretovaný*.

Java zajišťuje „neprodyšnou" *bezpečnost* – Java program nemá přístup k ničemu k čemu mu klientův počítač přístup umožní.

Java je *objektově orientovaná*. Její syntaxe podněcuje programátory ke generování modulárních, přehledných programů zaměřených, v duchu OOP, na data v aplikaci a na metody manipulujícími s těmito daty.

Další velmi významnou vlastností je podpora *paralelního programování*. Vestavěná *podpora vláken* (threads) je jednou z nejmocnějších vlastností jazyka, umožňující programátorům vytvářet vysoce výkonné Java aplikace, které jsou díky paralelnímu

zpracování citlivější na uživatelovy požadavky, rychlejší a mnohem jednodušeji řiditelné.

Nezanedbatelnou výhodou využití vláken je také zvýšení *robustnosti* jazyka. Java je ovšem robustnější i z jiných důvodů: je silně typová (více než C++), vyžaduje explicitní deklaraci metod, nepodporuje ukazatele, má velmi promyšlený paměťový model (garbage collection) a metody ošetření/zpracování výjimek (exceptions).

Java byl navržen tak, aby se dokázal přizpůsobit vyvíjejícímu se prostředí. Je to tedy jazyk *dynamický*. Java tak například zavádí třídy podle potřeby a to i po síti.

Navíc je Java *jednoduchý* a *výkonný* jazyk, který pro nováčky vypadá Java jako jiný objektově-orientovaný jazyk, C++. Nepochybně je Java značně ovlivněna svým předchůdcem. Nicméně podobnost jazyka Java a C++ je pouze v syntaxi a Java se velmi liší od jazyka C++ v mnoha základních myšlenkách. V žádném případě nemůže být Java považována za „ořezaný" jazyk C++, jak se domnívá mnoho „zásadových" programátorů v C++.

Jedním z hlavních cílů jazyka C++ byla zpětná kompatibilita s jazykem C, který nebyl vůbec objektově orientovaný. Jazyk C++ je extrémně komplexní a propletený, což je daň *za kompatibilitu* s jazykem navrhovaným před skoro 30 léty. Jazyk C++ tím také zabřednul do některých syntaktických a implementačních bažin svého předchůdce (vzpomeňme například vícenásobnou dědičnost).

Návrháři jazyka Java se rozhodli snížit kompatibilitu s jazykem C a tím dosáhnout určitého zjednodušení. To umožnilo jazyku Java přijmout mnohem jednodušší, více *konzistentní styl*, *něčím připomínající* Smalltalk. (Pokud někdo není obeznámen s jazykem C++ nebo Smalltalk, vůbec se nemusí ničeho obávat – znalost těchto jazyků není pro práci s jazykem Java vůbec nutná.)

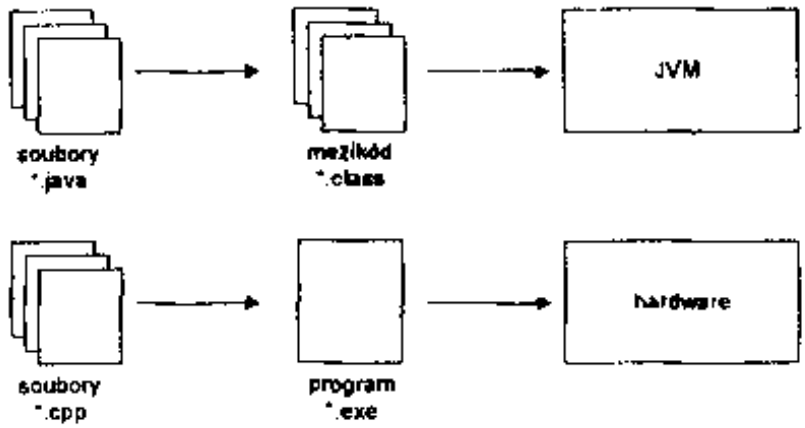
3.1 Jazyk Java

Java je *téměř* čistý objektově-orientovaný jazyk. Je však nutno přiznat, že ne vše je objekt. Všechen kód Java je udržován v metodách tříd. Všechny stavy jsou udržovány v atributech tříd. Všechny třídy s výjimkou **Object** dědí rozhraní (interface) a implementaci z právě jedné jiné třídy. Jedinými ne-objektovými typy jsou primitivní typy (boolean, byte, short, int, long, float, double, ...) a typ rozhraní. Objekty jsou buď instancemi třídy nebo polí. Objekty mohou být vytvořeny, ale ne explicitně *dealokovány* (!); *Java používá* automatický garbage collector zvyšující bezpečnost. Třída může implementovat libovolný počet rozhraní a *rozhraní* může být ve vícenásobné dědičné relaci s jinými rozhraními. Všechny Java třída a rozhraní přísluší určitému balíku (packages). (Od verze Java 1.1, může být třída také vložená do jiné třídy, nebo metody - Inner Classes).

3.2 Jak Java funguje

Pro pochopení filosofie jazyka Java je nutné objasnit některé pojmy použité při stručné charakteristice jazyka a doplnit charakteristiku o část „jak to vlastně funguje"

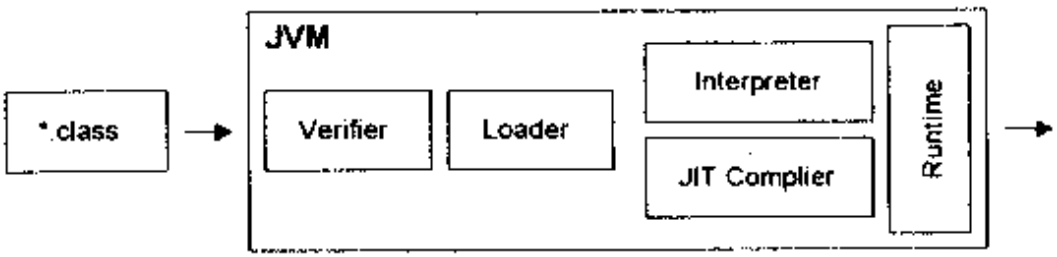
K čemu tedy dojde, napíšeme li program pro jazyk Java (uložený v souboru *.java). Prvním krokem k jeho úspěšnému provedení je jeho kompilace. Na rozdíl od konvenčních programovacích jazyků, jejichž kompilátor vygeneruje spustitelný kód (obsahující například instrukce 80x86), vygeneruje kompilátor Java (sám napsaný v jazyce Java) jakýsi „mezikód" nazývaný *Java Byte Codes* (soubory *.class). Mezikód jsou (optimalizované) instrukce napsané pro nějaký virtuální stroj Java, který reálně neexistuje. Pro spuštění našeho programu musíme mít k dispozici interpreter mezikódu který provede instrukce mezikódu, což znamená, že ve skutečnosti emuluje nějaký virtuální stroj Java (JVM – Java virtual machine) na libovolném počítači a operačním systému.



Obr.1. Kompilací zdrojových souborů *.java vznikají soubory *.class mezikódu obsahující instrukce Java Byte Codes, které jsou vykonány JVM.

Existence JVM má vliv zvláště na následující vlastnosti jazyka:

- *Platformová nezávislost*: Existuje pouze jeden jediný JVM (ve skutečnosti neexistuje žádný – proto je virtuální). Ovšem pro každou výpočetní platformu je vytvořen samostatný emulátor JVM (programovaný v C++). To poskytuje jazyku Java jeho platformovou nezávislost.
- *Bezpečnost*: JVM zajišťuje mimořádnou bezpečnost systému před neoprávněným zásahem zvenčí. JVM se k otázce bezpečnosti staví tak, že „ničemu se nedá věřit". Každý mezikód načítaný do JVM prochází verifikátorem, který ověří jeho neporušenost (konzistenci), zda neobsahuje některá jazyková omezení jazyka a další nejen systémové vlastnosti (jako například digitální podpis). Navíc mezikód vykonávaný instrukci po instrukci interpreterem je pod neustálým dohledem a nemá přístup k žádným systémový zdrojům klientského počítače k nimž mu JVM přístup neumožní (platí pro applety, u samostaných aplikací není toto bezpečnostní opatření aplikováno).
- *Redukce velikosti programů*: Java generuje extrémně malé spustitelné soubory, čímž napomáhá rychlejšímu natažení přes potenciálně pomalou komunikační linku. Java toho dosahuje velmi elegantním způsobem. (Mimo jiné) všechny „systémové" knihovny funkcí (API) jazyka jsou rezidentní v JVM namísto, aby byly linkovány do výsledného programu. Navíc JVM načítá pouze ty třídy které nezbytně potřebuje (mezikódy generované pro danou třídu nejsou nataženy, dokud není jejich přítomnost vyžadována).



Obr.2. Virtuální stroj Java (Java virtual machine)

Just-in-time kompilátor (JIT) kompiluje mezikód před spuštěním do nativního kódu příslušného procesoru, čímž se dosahuje zrychlení běhu programů napsaných v jazyce Java. Interepretace mezikódu nemůže být nikdy tak rychlá jako provedení normálního programu v nativním kódu. Ve skutečnosti je Java asi 10-20 krát pomalejší než program napsaný v C++. Samotný formát mezikódu je již optimalizován tak, aby byl proces generování nativního kódu JIT dostatečně rychlý, jednoduchý a dával poměrně dobrý kód (který je rychlostí srovnatelný s programem napsaným v C++). Samozřejmě, že v JVM je umožněno smíšené zpracování (kompilované a interpretované zpracování).

4. Java a databázové technologie

4.1 JDBC - Java Database Connectivity

Java API standardně poskytuje přístupový interface ke standardním SQL databázím. Interface zajišťuje jednotný přístup k širokém řadě relačních databází. Také poskytuje obecný základ na němž mohou být vybudovány high-level nástroje a interface.

S využitím jazyka Java a JDBC je jednoduché odeslat SQL zprávu (SQL statements) na vlastně libovolnou relační databázi. Jinými slovy, s Java a JDBC API není nezbytné napsat jeden program pro přístup do Sybase databáze, jiný program pro přístup do databáze Oracle a další pro databáze Informix. Stačí napsat jediný program a tento bude schopen odeslat SQL zprávu na příslušnou databázi. Navíc, s aplikací napsanou v jazyce Java, si nemusíme dělat starosti s psaním různých programů pro různé výpočetní platformy. Tedy kombinace Java a JDBC dovoluje programátorům vytvářet aplikace v duchu hesla jazyka Java „write it once and run it anywhere".

Navíc, pokud připočteme další vlastnosti jazyka – robustnost, bezpečnost, jednoduchost použití, snadné pochopení a možnost automatického natažení přes síť zjistíme, že je Java znamenitým jazykem pro tvorbu databázových aplikací.

5. Java a tvorba systémů z komponent

Než začneme hovořit o komponentách, musíme zodpovědět otázku „Co je komponenta?". Komponenta je malá, dobře navržená mini-aplikace která vykonává specifické úlohy. Vývojář může sestavit nějakou aplikaci spojením několika samostatných komponent; dokonce vytvořených v různých programovacích jazycích.

Použití komponent velmi rozšiřuje OO koncepci, schopnost definovat diskrétní části programů a propojit je do jedné výsledné aplikace. Což umožňuje využitím množiny komponent dosáhnout konzistentního návrhu a sdílení kódu mezi aplikacemi.

Komponenty jsou v současnosti velmi moderní a oblíbené. Existuje několik programovacích jazyků založených na komponentách (Delphi). Také Java obsahuje API umožňující využívat a vytvářet komponenty. Komponenty mohou být různých typů a funkcí. Některé mohou vytvářet uživatelské rozhraní – například tlačítka a přepínače, jiné mohou být „neviditelné" komponenty, jako časovače a analyzátory (separátory). Komplexní komponenty mohou definovat spustitelnou aplikaci, jako je například tabulkový procesor, nebo kalkulátor, která v mnoha případech sama obsahuje nebo rozšiřuje jiné komponenty – postupy, které jsou jádrem objektově-orientovaného programování.

5.1 JavaBeans a Active X

Verze jazyka Java před verzí 1.1 umožňovaly vytvářet pouze dva základní produkty: applety a samostatné aplikace. Ani jeden nemůže označovat komponentu – applet je skutečně jenom mini aplikace spuštěná prohlížečem, nebo jiným prostředím. Ačkoliv applety mohou být uspořádány tak, aby na Web stránce vystupovaly společně, neexistuje žádný způsob, jak by spolu mohly komunikovat. Každá jejich vzájemná interakce musí být provedena na straně serveru. Instance appletu je tedy velmi izolovaná od čehokoliv spuštěného ve stejném prostředí. Applet je tedy pouze degenerovanou komponentou, protože nemůže najednou přijít se všemi zdroji, dokonce ani se všemi třídami, které jsou nutné pro spuštění. Namísto toho, musí všechny zdroje a další třídy natahovat jednu za druhou ze serveru hostící soubory appletu.

JavaBeans, uvolněné v říjnu 1996, se snaží vyplnit prostor a vytvořit nový typ produktu: Java komponenty, nazývané „beans" (JavaSoft, 1996). Je neštastné, že čistý rozdíl mezi třídou a objektem v jazyce Java nepřetrvává také v JavaBeans. Ačkoliv je bean skutečnou komponentou (množina tříd a zdrojů), jeho uživatelsky přizpůsobené a připojené instance jsou také nazývány beans. To je dosti zmatečné. Proto označení *bean* ukazuje na komponentu, a *bean instance* na objekt komponenty. Označení „bean object" by mohl být také zavádějící, protože bean obvykle sestává z mnoha Java objektů.

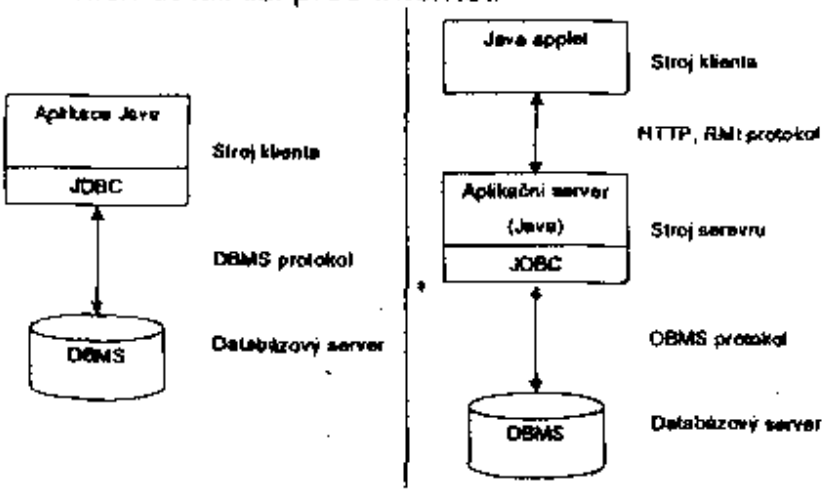
JavaBeans API definuje high-level software model komponent pro jazyk Java umožňující třetím osobám vytvářet a používat Java komponenty, které mohou být společně komponovány do aplikací konečným uživatelem. JavaBeans byl navržen tak, aby byl kompatibilní s ostatními standardy komponent zahrnujíce Active X a OpenDoc. Na rozdíl od technologie Active X firmy Microsoft, která je založena na binárním standardu meziobjektové komunikace COM (a je závislá na platformě Windows), je technologie komponent JavaBeans definována na JVM: tvůrci technologie soustředili pozornost na jazykovou nezávislost a později i na platfomovou nezávislost. V mnoha ohledech jsou Axtive X a JavaBeans kompatibilní. Každý může být definován v pojmech toho druhého, ačkoliv oba standardy mají mírně odlišný pohled na komunikaci komponent a jejich vývoj.

Pro bližší a detailnější pohled na problematiku komponentního programování doporučuji všem zájemcům čerstvou knihu [SZYP98].

6. Java a INTERNET a INTRANET

6.1 JDBC a Internet

V části „Java a databázové technologie" jsme se zmínili o JDBC, rozhraní rozšiřující možnosti nasazení jazyka Java v oblasti databázových aplikací. Zde uvedeme příklady, které názorně ukáží možnosti praktického využití jazyka pro tyto aplikace. S jazykem Java a jeho JDBC API je například možné vytvořit (a publikovat) Web stránku obsahující applet (malý program v jazyce Java spustitelný v prohlížeči Web stránek) který využívá informace získané ze vzdálené databáze. Nebo můžeme vytvořit aplikaci která dovolí nějakému podniku připojit všechny své zaměstnance (dokonce i když pracují na Windows, Macintosh, a UNIX strojích) na jednu nebo více internich databází přes Internet.



Obr.3. Dvouvrstvý a třívrstvý model přístupu k databázi.

Kombinace Java a JDBC umožňuje šířit informace jednoduše, rychle a ekonomicky. Tato kombinace *umožňuje trvalé* využíváni instalovaných databází a přístup k informacím dokonce i když jsou informace uloženy v různých databázich s různou správou řízení. Výrazně zkracuje čas pro vývoj nových aplikací Zjednodušuje instalaci a správu verzí programů, protože programátor může napsat nějakou aplikaci, nebo aktualizovat již existující, umístil ji na server a v tu chvíli má každý *přístup* k nejnovější verzi.
Pro detailnější informace o produktech založených na JDBC odkazuji zájemce na [URL1]

6.2 JavaScript

JavaScript je interpretovaný, objektově-*založený* (object-based) script jazyk pro HTML dokumenty. A ačkoliv má menši možnosti než plně kvalifikovaný objektově-orientovaný jazyk jako C++ a Java, JavaScript je více než dostatečně výkonný pro určené účely – dodání dynamiky statickým HTML stránkám, propojení HTML stránky a vložených komponent, příp. Java appletů atp.

JavaScript není „ořezaná" verze žádného jazyka (je pouze vzdáleně a nepřímo spřažený s jazykem Java a není ani nijak zjednodušená. Má ovšem určitá omezení. Nemůžou v něm být například napsány žádné samostatné aplikace a má pouze malé možnosti jak číst a zapisovat do souborů. Navíc může script napsaný v JavaScript běžet pouze za přítomnosti nějakého interpreteru buď na Web serveru, nebo v Web prohlížeči.

JavaScript je volně typový jazyk. To znamená, že nemůže explicitně deklarovat datové typy proměnných, ovšem JavaScript provádí v mnoha případech automatickou konverzi kdykoliv je to nutné. Například pokud se pokusíme vložit číselnou hodnotu do nějaké položky která je založena na textu (string) je číslo konvertováno na text. JavaScript má pouze tyto základní typy: number, boolean, string a null.

S ohledem na použití JavaScript ve Web prohlížečích, obsahuje jazyk některé předdefinované objekty spjaté s prohlížeči: window, location, history, document, Math, String, Date, ...

V současné době je JavaScript implementován ve všech základních Web prohlížečích: Netscape Navigator a Microsoft Explorer. Firma Microsoft vytvořila svou vlastní mutaci JavaScript, kterou nazvala JScript a který plně implementuje specifikaci jazyka podle ECMA 262 a navíc obsahuje některá rozšíření a je optimalizován pro Microsoft Internet Explorer, přičemž využívá některých jeho schopností.

Tab. 1. JavaScript versus Java

JavaScript	***Java***
Interpretovaný	Kompilovaný do byte kódu
Pouze dynamické typy	Statické a dynamické typy
Unifikované numerické typy	Hierarchie numerických typů
Objektově založený	Objektově orientovaný
Pomalý	Rychlý (rychlejší)
Žádné balíky (packages)	Balíky
Žádné třídy a interface	Třídy a interface

7. Podpora jazyka Java

V současné době je Java podporována snad všemi významnými výrobci software a každý kdo má k Java co říci vyvíjí a nabízí své vlastní „grafické" vývojové prostředí. Mezi hlavními jmenujme Sun Microsystems (Java Workshop), Microsoft (Visual J++) [URL2], Borland (JBuilder), Symantec (Café). Samozřejmě „referenčním" vývojovým nástrojem je stále Java Development Kit (JDK) [URL3] od Sun Microsystem [URL4], které je volně šiřitelné (prozatím??). Vývojáři Sunu si snad ani neuvědomili (či spíše nemohli tušit) co způsobí tím, že své vývojové prostředí začnou volně šířit. Volná šiřitelnost a vlastnosti jazyka způsobily ono nepředvídatelné a rychlé rozšíření jazyka a je pravděpodobně komerčním zájmem, aby toto základní prostředí bylo volně

šiřitelné. Dokonce software gigant Microsoft zakoupil licenci na jazyk Java a v současné době nabízí své vlastní (volně šiřitelné) vývojové nástroje (optimalizované pro prostředí Windows 95/NT) v software balíku nazývaném Software Development Kit for Java (SDK) [URL5].

8. Závěr

Java se také stává velmi často používaným jazykem nejen v oblasti výzkumu, ale i praktického nasazení (například Java Web Server firmy Sun). Java nabízí velké možnosti ve všech oblastech tvorby software a její prudký rozvoj (a vývoj) bude zajisté pokračovat i v následujících letech. Je oprávněné předpokládat, že bez alespoň základních znalostí koncepce tohoto jazyka se v blízké budoucnosti neobejde nikdo vytvářející počítačové aplikace, protože se s tímto jazykem budeme setkávat stále častěji a to nejen v podobě programů, ale i nových operačních systémů orientovaných na komponenty a síť.

Literatura

[DAVI96] Davis, S. R.: Learn Java Now, Microsoft Press, 1996

[FLAN97] Flanagan, D.: Programování v jazyce JAVA, Computer Press, Praha, 1997, ISBN 80-85896-78-8

[FLANA96] Flanagan, D.: Java in a Nutshell, O'Reilly & Associates, Inc., 1996

[LADD97] Ladd, S. R.: Active Visual J++, Microsoft Press, 1997, ISBN 1-57231-609-8

[LEA97] Lea, D.: Concurrent Programming in Java - Design Principles and Patterns, Addison-Wesley Longman, Inc., Reading, Massachusetts, 1997, ISBN 0-201-69581-2

[MICR97] Microsoft Visual J++ 1.1, Microsoft Corporation, 1997

[MICR97a] Microsoft SDK for Java Version 2.0 Documentation, Microsoft Corporation, 1997

[SUN97] The Java Language Specification, Sun Microsystems, Inc., California, 1997

[SUN97a] Java Development Kit 1.1.4 Documentation, Sun Microsystems, Inc., California, 1997

[SUN97b] The Java Tutorial: Object-Oriented Programming for the Internet, Sun Microsystems, Inc., California, 1997

[SZYP98] Szyperski C.: Component Software - Beyond Object-Oriented Programming, Addison-Wesley, Essex, 1998, ISBN: 0-201-17888-5

[TEMP97] Templ, J.: Java - The Best of Two Worlds?, Join Modular Language Conference JMLC '97, Johannes Kepler University Linz, Schlo Hagenberg, 1997

URL

[URL0] http://kat455.vsb.cz

[URL**Chyba! Nenalezen zdroj odkazů.**] http://java.sun.com/beans/index.html

[URL1] http://java.sun.com/products/jdbc/

[URL2] http://www.microsoft.com/visualj/

[URL3] http://java sun.com/products/jdk/1.1/

[URL4] http://java sun.com/

[URL5] http://www.microsoft.com/java/