3. vydání

Pavel Ponec

učebnice jazyka JAVA

a tvorba webových aplikací pro samouky

Učebnice jazyka Java a tvorba webových aplikací pro samouky
Pavel Ponec
https://jbook.ponec.net/cs3

Korektura prvního vydání: Google Gemini 1.5 Pro
Odborná korektura druhého vydání: Ing. Ondřej Guth, Ph.D.
Jazyková korektura druhého vydání: Ing. Vladimíra Krejčíková
Ilustrace obálky: Marie Brogowski
Vnitřní ilustrace: autor knihy (pokud není uvedeno jinak)

Vydavatel: Pavel Ponec, Brno
3. přepracované vydání (1. a 2. vydání vyšlo pod názvem Učebnice jazyka Java na webových příkladech pro úplné začátečníky)
2026, leden
ISBN 978-80-909828-0-2

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

PROPOJÍME TĚ SE SVĚTEM IT

Kvalitní vzdělání ve všech oblastech informatiky od informačních systémů a webových technologií přes počítačovou grafiku až po hardware.

PROČ JÍT NA FIT?

- 100% pokrytí všech oblastí IT
- více jak 200 firem pro praxi při studiu
- 100% zaměstnanost na trhu práce
- 19 špičkově vybavených laboratoří
- 12 výzkumných skupin

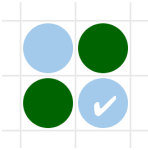
BAKALÁŘSKÝ STUDIJNÍ PROGRAM

- titul bakalář (Bc.)
- standardní doba studia 3 roky
- 10 specializací

NEPOTŘEBUJEŠ PŘEDCHOZÍ ZNALOST IT. VŠE TĚ NAUČÍME.

Poděkování

Rád bych poděkoval své laskavé dceři za mnoho užitečných podnětů a připomínek, bez kterých by tato kniha neměla současnou podobu. Poděkování patří také všem mým dlouhodobým kolegům v týmu za celou řadu užitečných rad, nápadů a konzultací. Kniha byla vytvořena pomocí open-source aplikací včetně ukázkových příkladů, vektorových obrázků a nástrojů pro sestavení výsledku – to vše na svobodném operačním systému Linux. Tímto bych rád vyjádřil své uznání a poděkování také široké komunitě lidí, kteří se starají o vývoj, údržbu a distribuci open-source projektů. V neposlední řadě děkuji také doc. RNDr. Josefu Kolářovi, CSc., a Ing. Ondřeji Guthovi, Ph.D., z Katedry teoretické informatiky FIT ČVUT za pomoc při druhém vydání této knihy.



Obsah

Poděkování . . . 1
1 Předmluva ke druhému vydání . . . 7
2 Úvodní informace . . . 8
2.1 Ochranné známky . . . 8
2.2 Vyloučení odpovědnosti . . . 8
2.3 Licence ukázkových příkladů . . . 9
2.4 Typografie . . . 9
2.5 Různé . . . 9
3 Úvod a motivace . . . 10
3.1 Proč vznikla tato kniha? . . . 10
3.2 Co budeme potřebovat? . . . 11
3.3 Proč se učit programovat? . . . 11
3.4 Proč se učit právě jazyk Java? . . . 12
3.5 Proč webové aplikace? . . . 13
3.6 Pár slov o jazyku Java . . . 13
4 Základní pojmy . . . 15
4.1 Data . . . 15
4.2 Algoritmus . . . 15
4.3 Paměť . . . 18
4.3.1 Ukládání čísel . . . 18
4.3.2 Ukládání znaků . . . 19
4.3.3 Ukládání logických hodnot . . . 19
4.3.4 Jak počítač rozumí informacím v paměti . . . 19
4.4 Proměnná . . . 20
4.5 Primitivní datové typy a přetypování . . . 20
4.6 Java výrazy a příkazy . . . 22
4.7 Stromová struktura dat . . . 24
4.8 Model reálného světa . . . 25
4.9 Fyzikální objekt . . . 25
4.10 Programový objekt . . . 26
4.11 Třída objektů . . . 26
4.11.1 Diagram modelu tříd . . . 27
4.11.2 Název třídy . . . 28
4.11.3 Atributy . . . 28
4.11.4 Metody . . . 28

4.11.5 Konstruktor . . . 29
4.11.6 Třída ve zdrojovém kódu . . . 29
5 Od třídy k reálnému kódu . . . 31
5.1 Užití objektu . . . 31
5.1.1 Texty v jazyce Java . . . 31
5.1.2 Vytvoření instance a užití třídy . . . 31
5.1.3 Autoboxing . . . 33
5.1.4 Neplatné objekty null . . . 34
5.1.5 Vztahy mezi třídami . . . 35
5.2 Další poznámky ke třídám . . . 38
5.2.1 Viditelnost metod, atributů a tříd . . . 38
5.2.2 Dědičnost . . . 39
5.2.3 Výjimky . . . 42
5.2.4 Balíčky . . . 45
5.2.5 Komentáře kódu a JavaDoc . . . 46
5.2.6 Interface . . . 48
5.2.7 Generické datové typy . . . 49
5.2.8 Inline třídy a lambda výrazy . . . 50
5.2.9 Třída typu Record . . . 51
5.2.10 Knihovny . . . 52
5.3 Datový typ pole . . . 52
5.3.1 Pole vytvořené metodou třídy . . . 53
5.4 Vybrané třídy standardní knihovny . . . 54
5.4.1 Třída String . . . 54
5.4.2 Rozhraní List . . . 55
Třída ArrayList . . . 56
List s typem prvku . . . 57
5.4.3 Rozhraní Map . . . 57
5.4.4 Rozhraní Stream . . . 58
5.4.5 Třída BigDecimal . . . 59
5.4.6 Shrnutí kapitoly o třídách . . . 61
6 Webové technologie a pojmy . . . 62
6.1 XML – rozšiřitelný značkovací jazyk . . . 63
6.2 HTML – hypertextový značkovací jazyk . . . 65
6.3 CSS – Kaskádové styly . . . 66
6.4 URL – Adresa internetové stránky . . . 69
6.5 JSON – úsporný datový formát . . . 70

6.6 Shrnutí . . . 71
7 Řešené příklady . . . 72
7.1 Instalace . . . 73
7.1.1 Hardware a operační systém . . . 73
7.1.2 Apache Maven . . . 74
7.1.3 Znakový terminál . . . 75
7.1.4 Získání příkladů . . . 75
7.1.5 Instalace vývojového prostředí . . . 75
7.1.6 Spouštění příkladů . . . 76
7.1.7 Co dělat při potížích? . . . 76
7.1.8 Virtuální počítač . . . 77
7.1.9 Virtualizační kontejner . . . 78
7.1.10 Vývojové editory . . . 79
7.2 Servlety . . . 79
7.2.1 Hello, World! . . . 80
První servlet . . . 81
Výsledek servletu . . . 84
7.2.2 Datový model HTML stránky . . . 84
Připojení knihovny do projektu . . . 87
7.2.3 Programové výrazy . . . 88
7.3 Tvorba tabulek . . . 90
7.3.1 Jednoduchá tabulka . . . 90
Příkaz for . . . 91
Inkrementace a dekrementace . . . 92
Řešení úkolu . . . 93
Alternativní procházení pole . . . 94
Podmíněný příkaz if-else . . . 95
Tabulka náhodných čísel . . . 95
7.3.2 Přehled vozidel . . . 97
7.4 Formuláře . . . 99
7.4.1 Formulář s jedním vstupním elementem . . . 99
7.4.2 Formulář s více vstupními elementy . . . 101
Regulární výrazy . . . 102
Formulář osobních údajů . . . 103
Ternární operátor . . . 106
Hodnocení platnosti textů . . . 107
7.4.3 Počítání slov . . . 108

7.5 Zábava . . . 110
7.5.1 Vlastní rodič servletu . . . 110
7.5.2 Bodové kreslení . . . 111
Pomocná třída Base64Converter . . . 113
Parsování textu . . . 113
Class – třída pro popis tříd . . . 114
Logování událostí . . . 114
Třída Optional . . . 115
7.5.3 Binární podoba textu . . . 115
Řešení . . . 116
7.5.4 Piškvorky s defenzivní strategií hry . . . 118
Diagram tříd . . . 119
Jak to funguje? . . . 119
Enumerátor . . . 121
Příkaz switch . . . 123
Popis tříd diagramu a jejich metod . . . 124
Zpracování dotazu . . . 125
7.6 Interaktivní aplikace s AJAX . . . 126
7.6.1 Co je to AJAX . . . 127
7.6.2 Testování regulárních výrazů . . . 128
7.6.3 Piškvorky s rychlejší odezvou . . . 131
7.7 Herní strategie od umělé inteligence . . . 134
7.7.1 Jak formulovat požadavek na herní strategii pro AI . . . 134
7.7.2 Závěrečné kroky a autorská práva . . . 135
7.7.3 Automatizované testy . . . 135
7.8 Verzování zdrojového kódu . . . 136
8 Úvod do databází a práce s daty . . . 139
8.1 Základní pojmy relačních databází . . . 139
8.2 Relace a entity . . . 140
8.3 Jazyk SQL aneb jak mluvit s databází . . . 141
8.3.1 Tvorba databázové tabulky . . . 141
8.3.2 Vložení řádku . . . 142
8.3.3 Čtení řádku . . . 143
8.3.4 Změna záznamu . . . 144
8.3.5 Mazání řádku . . . 144
8.3.6 Databázové transakce . . . 145
8.4 Programové rozhraní Javy . . . 145

8.4.1 Základní rozhraní JDBC . . . 145
8.4.2 Pokročilé rozhraní JPA . . . 146
8.4.3 Jak usnadnit práci s JDBC . . . 147
8.5 Hledání hotelů v jazyce Java . . . 147
8.6 Diagram tříd . . . 148
8.7 Životní cyklus databázového spojení . . . 149
8.8 Čtení databáze v Java objektech . . . 151
8.9 Vkládání Java objektů do databáze . . . 153
8.10 Generická abstraktní třída . . . 154
9 Jak psát dobrý kód . . . 156
Reference a zdroje pro další studium . . . 158
Rejstřík slov . . . 161

1 Předmluva ke druhému vydání

Kniha, kterou právě čtete, je úvodem do světa tvorby aplikací. Nečekejte však suchý popis programovacího jazyka. Čtenář se na jednoduchých příkladech naučí, jak při tvorbě programů přemýšlet. Pokud vás v této souvislosti straší slovo „algoritmizace“, kniha vás nezklame. Neobsahuje partie z teoretické informatiky, namísto toho je vše vysvětleno formou příkladů pochopitelných i pro naprosté nováčky ve světě tvorby software.

Tato publikace však návodem, jak při programování přemýšlet, nekončí. Čtenář se dozví, jak používat jazyk Java, jeho pravidla, a v neposlední řadě knihovny, které s tvorbou software pomohou. Tento jazyk je již dlouho osvědčenou volbou pro výuku základů programování, jeho znalost zároveň uplatníte i v praxi. Ani jazykem Java záběr knihy nekončí. V knize se dozvíte o základních technologiích webu a naučíte se tvořit webové aplikace.

Text zabíhá i mimo téma základů nutných pro vytvoření fungujícího programu. Dozvíte se, jak zdrojový kód organizovat, aby byl přehledný a pochopitelný, nebo proč a jak zdrojový kód obohatit o dokumentaci. Jistě oceníte popis aplikací, které s programováním pomohou, například editoru – vývojového prostředí. Seznámíte se i s dlouhodobě osvědčenou a stále používanou technologií, díky které je možné své softwarové projekty efektivně spravovat nebo sdílet s dalšími vývojáři. Kromě příkladů částí programu v textu je součástí knihy i hotová aplikace spolu s návodem na její zprovoznění. Vedle možnosti prohlédnutí všech detailů je tak možné se učit pomocí zkoušení si vlastních úprav.

Přeji příjemné vykročení do oblasti programování při čtení této knihy.

Ing. Ondřej Guth, Ph.D. v roce 2022

2 Úvodní informace

V této kapitole jsou uvedeny některé důležité informace, které však nemají přímý vliv na porozumění technickému výkladu.

2.1 Ochranné známky

Oracle, **Java** a **VirtualBox** jsou registrované ochranné známky společnosti Oracle Corporation anebo jejích poboček. Autor knihy nemá žádnou vazbu ke společnosti Oracle. **Apache Maven**, **Apache Tomcat**, **Apache NetBeans** a **NetBeans** jsou ochranné známky neziskové organizace Apache Software Foundation (ASF) [25]. **Ubuntu**, **Xubuntu** a **Canonical** jsou registrované ochranné známky společnosti Canonical Ltd. **Microsoft** a **Windows** jsou registrované ochranné známky společnosti Microsoft Corporation v USA anebo jiných zemích. **Docker** je registrovaná ochranná známka společnosti Docker, Inc., v USA anebo jiných zemích. Docker, Inc., a také další strany mohou mít práva k ochranné známce. **Eclipse** a **Jetty** jsou registrované ochranné známky společnosti Eclipse Foundation, Inc., ve Spojených státech a případně v dalších jiných zemích. **ECMAScript** je ochranná známka skriptovacího jazyka specifikovaná standardem ECMA international.

2.2 Vyloučení odpovědnosti

Mějme na paměti, prosím, že vzorové ukázky programového kódu byly vytvořeny za účelem výuky. Naznačují koncept, ale obsahují některá zjednodušení, například ošetřování vyhozených výjimek je z důvodu stručnosti omezené, a proto ukázky nejsou určeny přímo k produkčnímu použití.

I když autor věnoval značné úsilí odstraňování chyb a překlepů v textu, nelze vyloučit jejich přehlédnutí. Opravy, které byly zaregistrovány po vydání knihy, najdeme na její domovské stránce [1], kde lze nahlásit případné nesrovnalosti ve výkladu či jiné připomínky. Děkuji za všechny konstruktivní podněty.

2.3 Licence ukázkových příkladů

Po zakoupení knihy si prosím uschovejte doklad o nákupu, protože vás opravňuje používat ukázkové příklady k soukromým i komerčním účelům. Distribuce zdrojového kódu k této knize je možná pouze při zachování původního komentáře v hlavičce. Třetí strana je oprávněna použít zdrojový kód pouze pro své vlastní studium, jinak je třeba získat vlastní licenci zakoupením knihy nebo e-booku. Licence tak umožňuje pedagogům použití ukázkového projektu při výuce. V případě nejasností se obracejte na vydavatele knihy, nebo přímo na autora, kontakty najdete na internetové stránce knihy [1].

Zvažte, prosím, že výroba knihy není jedinou nákladovou položkou, další výdaje souvisí s propagací a distribucí. Děkuji tímto za podporu i pochopení.

2.4 Typografie

Slova psaná *kurzívou* budou označovat názvy produktů a nové technické termíny. Kvůli jejich vzrůstajícímu počtu však tyto termíny nebudou vyznačovány v celém rozsahu knihy, ale jen ve vybraných částech pro zdůraznění jejich specifického významu. Slova psaná `neproporcionálním` písmem (znaky s pevnou šířkou) značí programové *výrazy* a *příkazy*, klávesy nebo názvy souborů. V ukázkách zdrojového kódu jsou klíčová slova jazyka *Java* vyznačena modře (●), třídy objektů zeleně (●) a literály červeně (●).

2.5 Různé

Jména osob, uvedená v programových ukázkách v této knize, jsou smyšlená za účelem názorného výkladu a jejich případná podobnost s reálnými osobami je čistě náhodná.

3 Úvod a motivace

> *Write once, run anywhere.*
>
> — Sun Microsystems, americká společnost

Od roku 1991 usilovala firma *Sun Microsystems* o vytvoření jednoduchého, robustního, objektově orientovaného programovacího jazyka, jehož aplikace by byly nezávislé na platformě zařízení. Hlavní myšlenku reprezentoval slogan v záhlaví této kapitoly. Vedoucím projektu a původním autorem byl kanadský softwarový programátor James Gosling. Roku 1996 vydala firma první oficiální verzi jazyka pod názvem *Java* a o jedenáct let později (roku 2007) uvolnila většinu zdrojových kódů pod licencí **open-source**. Počátkem roku 2010 proběhla akvizice firmy *Sun Microsystems* společností *Oracle*, která se tak stala vlastníkem oficiální implementace Java SE platformy.

3.1 Proč vznikla tato kniha?

Když jsem s jazykem *Java* začínal, měl jsem pocit, že učebnice by se dala napsat i zábavnější formou. Po letech praxe jsem zveřejnil článek o tvorbě webových stránek v Javě pomocí objektového modelu HTML elementů. Napadlo mě, že právě to by mohl být skvělý základ pro interaktivní příklady při výuce. Když jsem pak v knihkupectví otevřel jednu z nových učebnic, znovu jsem narazil na formální popis, černobílé obrázky a ukázky postavené na zastaralé knihovně Swing. Proto jsem se rozhodl vykročit na nejistou dráhu autora inovativní učebnice programování. Výsledek máte právě před sebou.

Abych příklady co nejvíce přiblížil reálným aplikacím, ponechal jsem názvy programových konstrukcí (jako jsou *třídy*, metody či komentáře) v angličtině. Pro pochopení principů webových aplikací si vysvětlíme základy jazyka *HTML* (z anglického *Hypertext Markup Language*) a zmíníme i další technologie. Kniha má také přesah do oblasti návrhu a životního cyklu informačních systémů. Vzhledem k širšímu záběru témat se nebudeme zabývat všemi aspekty jazyka do nejmenších detailů. Místo toho se zaměříme raději na praktické využití, pochopení souvislostí a využití užitečných vlastností. Jsem si vědom, že ve snaze o čtivost textu jsem se občas dopustil určitého zjednodušení. Kniha tedy neposkytuje úplnou specifikai jazyka, ale nabízí spíše nadhled potřebný pro další samostatné studium. Výklad se záměrně vyhýbá konstrukcím, které jsou sice platné, ale v praxi považované za nevhodné či zastaralé. Důležitým doplňujícím zdrojem informací je dokumentace standardní knihovny zvaná *JavaDoc* [5]. Užitečné rady a řešení problémů lze najít také na serveru *Stack Overflow* [15]. Pokud se podaří probudit ve čtenáři zájem o vývoj aplikací, splní jeden ze svých hlavních cílů.

3.2 Co budeme potřebovat?

Kniha předpokládá, že čtenář je běžným uživatelem internetu, zvládne instalaci počítačových programů (například hry nebo internetového prohlížeče), orientuje se v základních pojmech, jako jsou například *pevný disk* či *soubor* na disku, a rozumí organizaci uložených *souborů* na disku. První kapitola se zaměřuje na motivaci výuky programovacího jazyka *Java* a zmiňuje také výhody webových aplikací.

Pro lepší pochopení obsahu knihy doporučuji zkoušet si jednotlivé příklady na počítači a následně zkoumat vliv vlastních úprav zdrojového kódu na chování spuštěné aplikace. V případě potíží je vždy možné aktuální projekt nahradit původní verzí jejím rozbalením z archivu.

Co tedy budeme potřebovat pro spuštění ukázkových příkladů?

1. Počítač (či notebook) s volnou kapacitou RAM **minimálně** 1 GB a 1 GB **volného** místa na pevném disku s operačním systémem *Windows*, *Linux*, *MacOS*, případně i jiným, na který lze nainstalovat *Java Development Kit*.
2. Připojení k internetu nejen pro instalaci potřebných programů, ale i k automatickému stažení nezbytných knihoven během spuštění přiložených příkladů.
3. Prostředí *Java Development Kit*, verze 25, pro sestavení a spuštění příkladů.

Další podrobnosti lze najít v kapitole O instalaci.

3.3 Proč se učit programovat?

Pokud se vám dostala tato kniha do ruky, zřejmě už jste nějakou motivaci našli. Pracovní trh se potýká s nedostatkem odborníků, a to nejen v oboru informačních technologií (dále jen IT). V mnoha oborech může být řešením nasazení automatizace či umělé inteligence, nic z toho se však neobejde bez vývojářů softwaru (SW), což ještě více posílí poptávku po IT odbornících. Běžným obchodníkům pomůže někdy i jednoduchá prezentace zboží (či služeb) na internetu s možností sestavení objednávky a následné platby.

Je dobré vědět, že zadání projektu i výsledek práce SW vývojáře se sdílí relativně snadno přes internet, a tak pokud někdo má dostatečné jazykové znalosti, a získá zároveň důvěru zákazníka, může dodávat svoji práci třeba na druhý konec světa z pohodlí svého domova.

3.4 Proč se učit právě jazyk Java?

Říká se s nadsázkou: „Kolik právníků, tolik názorů.“ U programovacích jazyků je to podobné: diskuze o výhodách různých programovacích jazyků bývají velmi emotivní, přitom každý argument může mít svůj díl pravdy. V této nepřehledné situaci může být pro začátečníka obtížné vybrat správný směr. Proč tedy *Java*?

1. Programovací jazyk *Java* patří na trhu práce dlouhodobě k nejžádanějším. Je tedy pravděpodobné, že čas strávený studiem se podaří brzy zhodnotit [6].
2. Hotové aplikace lze spouštět (při dodržení určitých pravidel) na různých operačních systémech (například *Windows*, *Linux*, *MacOS*), pokud je na nich k dispozici běhové prostředí zvané *Java Runtime Environment* (dále jen *JRE*). Takové schopnosti programovacího jazyka se označují výrazem multiplatformní. Standardní *Java* obsahuje také sofistikovanou knihovnu na tvorbu plnohodnotných desktopových aplikací pro podporované platformy.
3. Nástroje nezbytné pro vývoj a provoz standardních komerčních aplikací lze získat zdarma, včetně pokročilých editorů zaměřených na vývoj aplikací (dále jen IDE z anglického *Integrated Development Environment*), podporujících i sestavení projektu a jeho spuštění. Referenční implementace *Javy* s otevřenou licencí má název *OpenJDK* [4]. Při instalaci *Javy* však nejsme vázaní na jediného dodavatele, její implementaci nabízí několik dalších společností.
4. Vývojáři *Javy* mají k dispozici rozsáhlou nabídku různých **knihoven** s otevřeným zdrojovým kódem (anglicky *open-source*), které pokrývají široké spektrum nejrůznějších témat, což šetří vývojářům drahý čas i zbytečnou práci. Jedním tématem se někdy zabývá hned několik knihoven, které se liší svým programovým rozhraním či implementací.
5. Kolem jazyka se vytvořila silná komunita, která přispívá ke včasnému odhalení případných chyb či komplikací.
6. Vývojáři jazyka *Java* dosud věnovali značnou pozornost zpětné kompatibilitě jazyka při vydávání jeho nových verzí. V praxi to znamená, že uživatel má velkou naději, že spustí starší *Java* projekt i na poslední verzi prostředí (anglicky *run-time*). Vývojáři zase mohou používat (kompilovat) starý zdrojový kód bez úprav v poslední verzi překladače.
7. Jazyk nabízí silnou typovou kontrolu, která urychluje odhalení některých chyb zápisu zdrojového kódu, a tím šetří čas potřebný na testování aplikace.
8. *Java* se řadí k programovacím jazykům, které za běhu aplikace uvolňují automaticky paměť po nepotřebných objektech, což urychluje vývoj a eliminuje některé problémy s vyčerpáním paměti. Uvolňování paměti má na starost algoritmus označovaný anglickým termínem *Garbage collector*. V určitých situacích (například během řízení časově kritických procesů) však může krátké

časové prodlení spojené s uvolněním paměti představovat handicap, který lze ovšem eliminovat výběrem vhodné strategie algoritmu *Garbage collector*.

9. Programovací jazyk *Java* je k dispozici vývojářům relativně dlouhou dobu a dnes už je standardem pro výuku objektového programování (OOP). Znalost principů objektového programování jazyka *Java* může usnadnit pochopení objektového přístupu také v jiných jazycích.

3.5 Proč webové aplikace?

Webové aplikace pro internetové prohlížeče se těší stále větší oblibě koncových uživatelů i provozovatelů a důvodů je hned několik.

1. Internetové připojení je dnes už snadno dostupné pro většinu uživatelů osobních počítačů i mobilních zařízení.
2. Nové internetové technologie umožňují vývojářům stále zlepšovat ergonomii webového uživatelského rozhraní, které se pomalu blíží komfortu desktopových aplikací.
3. Webové aplikace jsou na operačním systému prakticky nezávislé, klienti tedy mohou používat libovolné zařízení, které poskytuje internetový prohlížeč.
4. Distribuce nových verzí SW produktů k zákazníkovi má minimální náklady.
5. Pro útočníka je obtížné webovou aplikaci odcizit, nebo prolomit její licenční omezení.

3.6 Pár slov o jazyku Java

Java je objektový programovací jazyk se silnou typovou kontrolou. Objektově orientované programování (dále *OOP*) je specifický způsob zápisu zdrojového kódu do textových souborů popisujících *třídy*. Ty mají svůj *název*, *datovou složku* a skupinu *algoritmů*, jež jsou obsaženy v pojmenovaných útvarech zvaných *metody*. Vlastnostmi *OOP* se budeme zabývat později, předtím si však ještě ujasníme některé základní pojmy z informatiky.

Zdrojový kód jazyka *Java* se zapisuje do textových souborů (s příponou `java`) podle pravidel, kterým se říká *specifikace jazyka* [3]. Jak se programovací jazyk vyvíjí, vznikají nové verze specifikace, které korespondují s určitým *Java* prostředím (instalovaným na počítači). Staré programy zpravidla fungují i v novějších prostředích *Javy*, této vlastnosti se říká *zpětná kompatibilita*. Příklady ke knize byly odladěny ve verzi *Java* 25, pravděpodobně je však bude možné používat (bez dalších úprav) i na verzích vyšších. Verze 25 *LTS* má dlouhodobou podporu.

Zdrojové soubory se ukládají do adresářové struktury *projektu* (ve *Windows* se souborovým *adresářům* říká *složka*). Stromová struktura projektu však není úplně libovolná, ale má svá pravidla – některá vycházejí ze specifikace jazyka. Více si řekneme v kapitole O sestavení aplikace.

Při instalaci *Javy* se setkáváme se dvěma pojmy:

1. *Java Runtime Environment* (JRE) je minimalistické běhové prostředí, které potřebuje koncový uživatel programu ke spuštění aplikace vytvořené v jazyce *Java*. Běhové prostředí některé společnosti ke stažení nenabízejí, vývojáři si ho však mohou vytvořit sami. Podrobnější informace lze najít třeba na odkazu [33].
2. *Java Development Kit* (JDK) je nástroj pro vývojáře, který poskytuje *kompilátor* zdrojových kódů a některé další služby užitečné pro vývoj a ladění aplikace. Prostředí *JRE* je automaticky obsaženo v *JDK*. Pro vývoj aplikací budeme potřebovat instalovaný produkt *JDK*.

Kompilátor v jazyce *Java* je program, jenž prověří správnou syntaxi (soubor pravidel) zápisu zdrojového kódu a převede ho do binární podoby (zvané *bytecode*), se kterou umí pracovat prostředí *JRE*. Kompilované soubory mají příponu `class`.

Obrázek 1. Znázornění kompilace souboru

Standardní kompilátor *JDK* se tím liší od některých jiných kompilátorů (i jiných jazyků), které převádějí zdrojový kód rovnou do binárního formátu, jenž obsahuje přímé instrukce procesoru počítače. Proces spuštění projektu si představíme v kapitole O sestavení aplikace.

Nezaměňujme programovací jazyk *Java* se skriptovacím jazykem *JavaScript*, protože se jedná o dva **různé** jazyky. Zdrojový kód *JavaScriptu* se provádí (interpretuje) bez předchozí kompilace, což neumožňuje zachytit včas překlepy a typové nekonzistence. Interpretace skriptu bývá pomalejší. Výhodou *JavaScriptu* je jeho podpora v internetových prohlížečích.

4 Základní pojmy

> *Pokud chcete problém vyřešit, musíte jej nejdříve pojmenovat, a řešit příčinu, ne příznaky.*
>
> — Otto Wichterle, český vědec a vynálezce

Většina problémů, se kterými se (nejen) začínající programátoři setkávají, má své řešení někde na internetu, stačí ho najít. Klíčem k nalezení řešení bývá správná formulace vyhledávacího dotazu, ale ten se bez správných pojmů sestavuje jen obtížně.

4.1 Data

V oblasti výpočetní techniky jsou *data* libovolné informace vhodné k počítačovému zpracování. Nejmenší jednotkou dat je *bit*, který může nabývat pouze dvou stavů: (**ano**/**ne**) nebo (**pravda**/**nepravda**) nebo třeba (**přítomen**/**nepřítomen**), interpretace mohou být různé. Protože stav lze vyjádřit také pouhou číslicí (`1`/`0`, anglicky *digit*), nazývají se různá zařízení (stroje, přístroje), která interně pracují s čísly, občas výrazem *digitální zařízení* (číslicové stroje, přístroje). Skládáním *bitů* do větších celků vznikají datové útvary, jejichž hodnota opět závisí na interpretaci. Pro vyjádření většího objemu dat se běžně používá jednotka *bajt* (anglicky *byte*) obsahující 8 *bitů*. U těchto jednotek můžeme použít předpony soustavy SI (*Mezinárodní systém jednotek*) podle vzoru:

- *kilobajt* – označuje tisíc *bajtů*, zkratka je *kB*,
- *megabajt* – označuje milion *bajtů*, zkratka je *MB*,
- *gigabajt* – označuje miliardu *bajtů*, zkratka je *GB*.

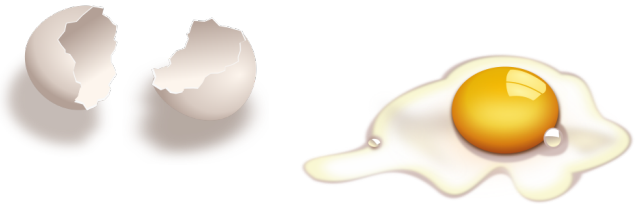
Předpony velkých čísel využijeme například při kontrole volného místa na pevném disku.

4.2 Algoritmus

Výklad pojmu *algoritmus* začneme jednoduchou definicí:

> *Algoritmus je návod či postup, kterým lze vyřešit daný typ úlohy. Přepis algoritmu do programovacího jazyka se nazývá programování.*
>
> — inspirováno zdrojem [10]

Jistou podobnost k *algoritmu* lze najít třeba v kuchařském receptu na přípravu pokrmu. Vezměme si například přípravu míchaných vajec pro dvě osoby.



Obrázek 2. Ilustrativní obrázek, upravený zdroj [9]

Postup přípravy lze rozepsat do následujících kroků:

1. Připravíme si suroviny: 4 vejce, šunku, olej a sůl.
2. Na pánvi ohřejeme olej na 160 °C.
3. Přidáme šunku nakrájenou na kostičky.
4. Přidáme vejce.
5. Mícháme po dobu 3 minut.
6. Pokrm podáváme na talíři teplý a osolený.

Tento recept jistě není zcela přesný, ale zkušená kuchařka ví, že vejce je třeba zbavit nejdříve skořápky a sporák je nutné nakonec vypnout; stroj to však neví. Z příkladu je také zřejmé, že pro srozumitelný popis přípravy pokrmu se neobejdeme bez jisté míry abstrakce. Hned první krok by se dal rozepsat samostatným algoritmem:

1. Vyhledáme v lednici potraviny na přípravu míchaných vajec.
2. Pokud některá z potravin chybí (kontrola obsahu lednice), pošleme člena domácnosti na nákup chybějících surovin.
3. Požadované suroviny odneseme na kuchyňskou linku.

Také **proces nakupování** by se dal popsat samostatným algoritmem a tímto způsobem bychom mohli pokračovat směrem ke stále podrobnějším detailům. Pojem *kroky algoritmu* lze chápat jako *příkazy*, které v jazyce Java realizujeme voláním metod. Každý příkaz přebírá odpovědnost za nějakou dílčí část algoritmu. Na našem příkladu vidíme, že součástí algoritmu mohou být také *příkazy* zpracované za určité podmínky (pokud nějaká potravina chybí, *běž nakoupit*).

Pokud pošleme člena rodiny na nákup, s dalším postupem přípravy pokračujeme až po jeho návratu. V běžném životě však zpravidla nečekáme, až se někdo vrátí, ale

třeba nachystáme jídelnu. Čtyřjádrový procesor si můžeme představit jako rodinu se čtyřmi členy a zpracování algoritmů několika jádry procesoru, které umožňuje *paralelní zpracování*. Toto téma však přesahuje plánovaný rámec knihy, a tak budeme dál uvažovat o všech algoritmech pouze v režii jediného jádra procesoru. Dopad na praktický život by to mělo ten, že nákup chybějících potravin zajistí kuchař sám.

Někdy nastane situace, že kuchař není schopen připravit požadovaný počet porcí najednou — třeba kvůli omezenému objemu kuchyňského nádobí. V tom případě můžeme postup vaření (*algoritmus*) opakovat tak dlouho, dokud počet porcí neodpovídá očekávání (*podmínka*). Procesu opakování se říká *iterace*. Protože však nakupování potravin, během každé iterace, by neúměrně zpomalilo čas přípravy, společný nákup lze provést jen jednou. Procesu urychlení pak můžeme říkat *optimalizace* algoritmu.

Programové *příkazy* se běžně provádějí shora dolů (nebo zleva doprava, pokud je více *příkazů* na jednom řádku). Každý *příkaz* se obvykle spustí až po dokončení předchozího. Pro tuto chvíli pomiňme příkazy jazyka pro *opakované* či *souběžné* zpracování na více procesorech.

Jak už jsme naznačili před chvílí, algoritmy se v jazyce *Java* zapisují (v textovém formátu) do útvarů zvaných *metody*. Každá metoda má svůj **název**, může vyžadovat seznam **parametrů** a může vracet návratovou hodnotu. Parametry umožňují *předávání dat* do metody z místa, kde je metoda volána svým jménem. Nákup potravin by vedl k návrhu metody s těmito vlastnostmi:

1. název metody: „běž nakoupit“
2. parametr metody: „seznam potravin“
3. návratový objekt: „nákupní taška se zakoupenými potravinami“

Z pohledu kuchařky je však popis *algoritmu* nákupu nezajímavý. Kuchařka potřebuje pouze vědět, jak se *metoda* pro zajištění surovin jmenuje a jak ji má použít.

Zapamatujme si, že při počítačovém zpracování se využívají *algoritmy* (například recept na přípravu pokrmů) a *data* (nákupní seznam, potraviny, přepravní taška). Průchod algoritmem lze (za nějaké podmínky) opakovat pomocí *iteračních* příkazů programovacího jazyka.

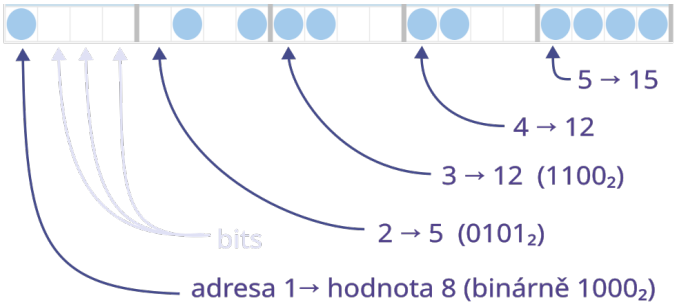
4.3 Paměť

> *Paměť je deník, který všichni nosíme s sebou.*
>
> — Oscar Wilde, irský dramatik

Počítačová *paměť* je místo, kam si počítač ukládá informace, které potřebuje k práci – krátkodobě nebo natrvalo. Díky paměti může počítač uchovávat data i instrukce, podle kterých má pracovat. Existují dva hlavní druhy paměti:

1. **Paměť RAM (Random Access Memory)** je krátkodobá, „pracovní“ paměť počítače. Ukládají se do ní data a programy, se kterými počítač právě pracuje. Když se zařízení vypne, obsah paměti RAM se vymaže. Aby mohl program paměť používat, musí ji nejdříve *alokovat* – tedy požádat operační systém o přidělení volného prostoru. Poté získá adresu paměti, kam může data zapisovat. Adresa je v podstatě pořadové číslo paměťové buňky, které určuje umístění informace. Po dokončení práce je nutné paměť uvolnit, aby ji mohl použít jiný program. Různé programy potřebují různé množství paměti – například kalkulačka využije méně než webový prohlížeč.
2. **Pevná paměť (např. pevný disk nebo SSD)** je trvalá paměť, která uchovává data i po vypnutí počítače. Jsou zde uložené všechny soubory, dokumenty, aplikace i operační systém počítače.

4.3.1 Ukládání čísel



Obrázek 3. Číselné hodnoty v paměti

Každý malý čtvereček na obrázku představuje jeden **bit**, který si pamatuje dva stavy: například „ano“ (má kuličku) a „ne“ (nemá ji). Pokud tyto stavy vyjádříme počtem obsažených kuliček (`1` nebo `0`), lze na spojení více bitů nahlížet jako na *dvojkovou číselnou soustavu*. Ve **čtyřech bitech** lze uložit 16 různých čísel (v rozsahu od `0` do `15` včetně). Kuličky z první adresy (`1`) pak můžeme přepsat do binárního čísla `1000`, které lze převést (třeba kalkulačkou) na číslo desítkové

Vážení čtenáři, právě jste dočetli ukázku z knihy Učebnice jazyka Java a tvorba webových aplikací pro samouky**.**
Pokud se Vám ukázka líbila, na našem webu si můžete zakoupit celou knihu.