EDICE
myslíme v ...

knihovna programátora

- Podrobný výklad vlastností jazyka od naprostých základů až po pokročilé, běžně neprobírané konstrukce
- Probírá i konstrukce, které budou zařazeny až do příštích verzí jazyka, a předvádí, jak tyto konstrukce vyzkoušet
- Vysvětluje nejenom jak probírané konstrukce používat, ale také proč jsou právě takové
- Využívá zabudované REPL prostředí pro demonstraci vykládaných konstrukcí bez zbytečného pomocného kódu
- Může sloužit současně jako učebnice i referenční příručka

RUDOLF PECINOVSKÝ

Java 14

Kompletní příručka jazyka

GRADA

Ing. Rudolf Pecinovský, CSc. je absolventem *Fakulty Elektrotechnické ČVUT* z roku 1979. Titul CSc. získal v *Ústavu teorie informace a automatizace ČSAV* v roce 1983. Od počátku 80. let učí a publikuje, přičemž svůj výzkum soustředí především na oblast vstupních kurzů moderního programování pro naprosté začátečníky. V současné době učí na *Vysoké škole ekonomické v Praze*, na *Fakultě jaderné a fyzikálně inženýrské ČVUT* a na *Vysoké škole podnikání a práva*. Doposud mu vyšlo přes 60 knih, které byly přeloženy do pěti jazyků. Většina jeho knih je zaměřena na výuku moderního programování a na umění návrhu objektově orientované architektury.

knihovna programátora

RUDOLF PECINOVSKÝ

Java 14

Kompletní příručka jazyka

GRADA
Publishing

Rudolf Pecinovský

Java 14
Kompletní příručka jazyka

Vydala Grada Publishing, a.s.
U Průhonu 22, Praha 7
obchod@grada.cz, www.grada.cz
tel.: +420 234 264 401
jako svou 7620. publikaci

Odpovědný redaktor: Ivana Palasová
Návrh vnitřního layoutu Rudolf Pecinovský
Zlom Rudolf Pecinovský
Počet stran 578
První vydání, Praha 2020
Vytiskly Tiskárny Havlíčkův Brod, a.s.

ISBN 978-80-271-1605-8 (ePub)
ISBN 978-80-271-1604-1 (pdf)
ISBN 978-80-271-1369-9 (print)

Všem, kteří se chtějí něco naučit

Stručný obsah

Podrobný obsah

Část III Pokročilé objektové konstrukce 387

Úvod

Tato kniha seznamuje se čtrnáctou verzí jazyka *Java*. Soustředí se především na výklad vlastností jazyka a minimalizuje výklad probírající obsah standardní knihovny – tomu se budou věnovat další příručky z této série.

Komu je kniha určena

Kniha je určena všem, kteří se chtějí naučit jazyk *Java*. Nevyžaduje od čtenáře žádné předchozí znalosti programování a snaží se, byť stručně, vysvětlit vše, co je potřeba.

Naprostým začátečníkům bych sice doporučil, aby začali s některou z mých příruček úvodu do programování, v nichž nevysvětluji, jak se program zapisuje, ale soustředím se především na to, jak se vymýšlí. V těchto příručkách spolu navrhujeme architekturu programu. Zakódování navrženého programu (tj. jeho zápis v programovacím jazyce) přenecháváme generátoru kódu, který je integrální součástí použitého vývojového prostředí. K výkladu zápisu programu v kódu, na nějž se soustředí tato učebnice, pak přejdeme až v okamžiku, kdy složitost navrhovaných programů překročí možnosti onoho generátoru. V té době ale již mají účastníci kurzu základy objektově funkcionálního paradigmatu zažité a nedělají proto chyby, s nimiž zápasí studenti, kteří začali studiem syntaxe použitého programovacího jazyka.

Na druhou stranu nechci nikoho zrazovat, a proto jsem se v této knize pokusil důkladně vysvětlit i základní programové konstrukce a doporučené způsoby jejich využití, takže i naprostí začátečníci zde najdou všechny potřebné informace.

Jak už jsem řekl, kniha se soustředí na výklad jazyka. Tím, že výklad knihoven (např. práce s kolekcemi, datovody, soubory a datovými proudy, regulárními výrazy či vlákny) odložím do jiných příruček, ušetřím prostor, který mohu věnovat podrobnému výkladu těch vlastností jazyka, na které v jiných příručkách často nezbývá místo. Jejich neznalost ale vede k hodinám marného pátrání po skutečném původci vzniklé chyby a následným experimentálním změnám programu s myšlenkou: „Co kdyby to náhodou vyšlo?“

Koncepce výkladu

Značná část učebnic začíná definicí programu typu *Hello world*, což je koncepce převzatá z historické učebnice jazyka C vydané v roce 1978. Základní nevýhodou této koncepce je, že na počátku používá několik programových konstrukcí, které čtenáři vysvětlí až někde v průběhu dalšího výkladu.

Tehdy to dost dobře jinak ani nešlo. Od té doby se objevila řada nástrojů, které umožňují koncipovat výklad tak, aby se v něm používaly pouze konstrukce, které již byly vysvětleny. O to se pokouším i v této knize a používám doposud nevysvětlené konstrukce opravdu výjimečně pouze v situacích, kdy to výrazně zpřehlední výklad. Aby se mi to podařilo, používám k výkladu program *JShell*, který je standardní součástí vývojářské sady.

Knihu jsem se navíc pokusil napsat tak, aby mohla sloužit stejně dobře jako učebnice jazyka i jako referenční příručka, ve které by bylo možno v případě potřeby najít klíčové informace rychleji a snáze než na internetu. Pasáže vysvětlující jednotlivé programové konstrukce jsem se proto snažil jasně oddělit od pasáží probírajících teorii, jak řešit tu kterou třída problémů, a sloužících především začátečníkům, kteří ještě nemají s programováním žádné zkušenosti.

Rozdělení textu

Text je rozdělen do čtyř částí. První část začíná úvodem do prostředí *JShell*, které bude v celé první části využíváno ke spouštění demonstračních programů. Zbytek první části pak probírá algoritmické konstrukce a prostředky, které *Java* nabízí k jejich realizaci.

Druhá část se soustředí na základy objektově orientovaného programování. Začíná teoretickou kapitolou vysvětlující hlavní zásady objektově orientovaného paradigmatu. Po ní následují kapitoly postupně probírající základní objektové konstrukce a jejich účel.

Třetí část je věnována výkladu nadstavbových objektových konstrukcí včetně rozboru některých základních pravidel, jejichž nedostatečné pochopení dělá studentům často problémy. Závěr třetí části je věnován koncepci modularity. Modularita sice není součástí jazyka, ale platformy; nicméně její osvojení přináší výrazné zvýšení spolehlivosti vyvíjených programů a do jisté míry i efektivity jejich vývoje i následného provozu.

Čtvrtá část obsahuje přílohy. První z nich vás seznámí s omezeními jazyka *Java*, druhá pak ukáže, jak se v *Javě* navrhuje GUI. Nejprve představí nástroje použitelné pro superjednoduché GUI, pak principiálně naznačí, jak se navrhuje komplexní GUI.

Terminologie

Při zavádění české terminologie se bohužel potýkáme často s tím, že v anglické literatuře zavádějí mnozí autoři terminologii vlastní. V oblasti, kterou se má zabývat tato publikace, je terminologický guláš obzvlášť vydatný. Budu-li proto uvádět ve svém výkladu anglické ekvivalenty zaváděných termínů, budu používat terminologii zavedenou v oficiální referenční publikaci *Java® Language Specification – Java SE 14 Edition* ([5]), na níž se budu občas odvolávat zkratkou JLS.

Použité nástroje

Pro úspěšné studium knihy budete potřebovat několik programů, které musíte nejprve stáhnout a instalovat.

Vývojová sada JDK

Především budete potřebovat mít instalovanou vývojovou sadu pro *Javu 14* označovanou JDK, což je zkratka z anglického *Java Development Kit*. Stáhnete ji ze stránky http://www.oracle.com/technetwork/java/javase/downloads/. Jenom musíte před vlastním stažením nastavit přepínač u seznamu stáhnutelných souborů do polohy **Accept License Agreement**, protože jinak vás stránka daný program stáhnout nenechá.

Spolu s vývojovou sadou vřele doporučuji stáhnout i dokumentaci. Tu najdete na téže stránce, jenom musíte popojet trochu níže do sekce **Additional Resources**, ve které najdete podsekci **Java SE 14 Documentation**.

Vývojové prostředí *JShell*

Pro většinu výkladu budu využívat prostředí *JShell*, které je součástí JDK a se kterým vás seznámí úvodní kapitola. (Podrobnější seznámení s tímto prostředím najdete v příručce *Java 9 – JShell*.) Výhodou tohoto přístupu je, že součástí programů nemusí být nejrůznější vata, bez které byste standardní program v *Javě* nespustili. Navíc již nemusíte instalovat žádné další vývojové prostředí, takže nebudete muset zápasit s důsledky známé skutečnosti, že zvládnutí současných profesionálních vývojových prostředí je náročnější než zvládnutí základů jazyka.

V závěru knihy budu potřebovat vysvětlit některé konstrukce, které se v prostředí *JShell* dost dobře vysvětlit nedají. Až na to dojde, tak vás upozorním.

Samostatné vývojové prostředí

Od kapitoly *12 Vývojová prostředí a vytvoření aplikace* na straně 243 začnu vedle prostředí *JShell* používat i prostředí *NetBeans*. Budu v něm demonstrovat vlastnosti, které se projeví až u samostatně vyvíjených programů. Nelpím přitom na tom, abyste používali právě toto prostředí, ale doporučuji je těm, kteří ještě nemají s profesionálními vývojovými prostředími žádné zkušenosti.

Adresy, z nichž si můžete stáhnout některé z doporučovaných prostředí, najdete v pasáži *Nejpoužívanější IDE* na straně 244. Každopádně budete-li chtít experimentovat s vybraným profesionálním prostředím hned od počátku, nic vám v tom nebrání.

Doprovodné programy

Všechny doprovodné programy zmiňované a používané v textu najdete na stránce knihy na adrese http://knihy.pecinovsky.cz/62_j14lang. Zde jsou umístěny ZIP-soubory obsahující doprovodné programu nebo jejich odvozeniny.

Doprovodné programy jsou zpočátku skripty prostředí *JShell*, v druhé části to pak jsou projekty pro vývojové prostředí *NetBeans*, které lze snadno transformovat na projekty pro vaše oblíbené vývojové prostředí. Podrobnosti o struktuře doprovodných programů najdete stránce s programy.

Knihu jsem se snažil zalomit tak, aby všechny výpisy programů a komunikace s počítačem, které se vejdou na jednu stránku, byly vždy na jedné stránce a nelámaly se přes hranu stránky. Čtenářům papírové verze by se tak měly minimalizovat problémy při studiu doprovodných textů diskutujících obsah těchto výpisů.

Čtenáři elektronické verze jsou na tom s listováním hůře. Těm bych doporučil, aby si text knihy otevřeli dvakrát a umístili oba dokumenty vedle sebe. (Na většině současných počítačů s širokými monitory by to neměl být problém.) Pak lze mít v jednom dokumentu otevřenou stránku s výpisem a ve druhém dokumentu stránku s rozborem obsahu daného výpisu.

Druhou možností je, že si studovaný výpis vytisknete (v doprovodných programech najdete příslušné soubory i s čísly řádků výpisu v knize), budete se dívat do výpisu a na čtečce číst doprovodný text.

Předběžné definice nových konstrukcí

Od verze 10 bylo zavedeno vydávání nových verzí dvakrát do roka: na jaře (typicky v březnu) vycházejí sudé verze, a na podzim (typicky v říjnu) verze liché. Protože dotažení návrhu některých konstrukcí trvá delší dobu, bylo v zájmu maximalizace zpětné vazby od uživatelů zavedeno, že vydané verze podporují i konstrukce, které jsou teprve ve vývoji, a nedoporučuje se je proto používat ve finálních verzích aplikací.

Ve standardní instalaci JDK se můžete setkat se třemi typy rozpracovaných konstrukcí, funkcionalit a aplikací:

Předběžná funkce/konstrukce/vlastnost (Preview Feature[1])

Jako předběžná je označena konstrukce jazyka nebo funkce VM, která je plně specifikována, plně implementována, ale nemusí být ve své definitivní verzi. Má iniciovat zpětnou vazbu vývojářů, kteří se ji pokusí použít ve svých aplikacích.

Při standardním nastavení tyto budoucí novinky překladač ani běhové prostředí nepodporují. Jejich podporu překladačem, resp. virtuálním strojem, resp. programem `javadoc`, nastavíte zadáním parametru

```
--enable-preview
```

příslušného programu (překladače, virtuálního stroje, programu javadoc).

Experimentální funkce/konstrukce/vlastnost (Experimental Feature)

Experimentální funkce představují rané verze funkcí, většinou na úrovni VM. Tyto funkce mohou být neúplné nebo dokonce nestabilní. Ve většině případů potřebují povolit pomocí vyhrazených příznaků.

Experimentální prvek je považován za zhruba z 25 % „hotový“, zatímco předběžný prvek by měl být „hotový“ alespoň z 95 %.

Inkubační funkce/konstrukce/vlastnost (Incubating Feature, Incubator[2])

Inkubační funkce jsou experimentální API distribuované ve formě samostatných modulů se jmény s předponou `jdk.incubator`. Abyste s nimi mohli pracovat, musíte tyto moduly explicitně přidat.

Syntaktické definice a diagramy

V testu je uváděna řada nejrůznějších programových konstrukcí, jejichž zápis se řídí přesně danými syntaktickými pravidly. Většina učebnic možné způsoby zápisu zpravidla formálně nedefinuje, anebo používá syntaktické definice odvozené z Backusovy normální formy. Syntaktické definice jsou výhodné pro strojové zpracování, lidé se v nich často špatně orientují. Rozhodl jsem se proto zobrazovat syntaktická pravidla zápisu jednotlivých konstrukcí pomocí syntaktických diagramů.

Syntaktický diagram ukazuje, jak je možno zobrazovanou konstrukci zapsat. Pojedete-li po čarách, tak jakýkoliv průjezd generuje syntakticky správnou konstrukci. Toho, kdo syntaktické diagramy ještě nezná a chtěl by rychle některý vidět, bych odkázal např. na diagram na obrázku 2.1 na straně 56.

---

[1] Podrobněji je koncepce předběžných vlastností popsána v JEP 12, které najdete na adrese https://openjdk.java.net/jeps/12.

[2] Podrobněji se o koncepci inkubačních modulů dozvíte v JEP 11, které najdete na adrese https://openjdk.java.net/jeps/11.

Použité typografické konvence

K tomu, abyste se v textu lépe vyznali a také abyste si vykládanou látku lépe zapamatovali, používám několik prostředků pro odlišení a zvýraznění textu.

Termíny První výskyt nějakého termínu a další texty, které chci zvýraznit, vysazuji **tučně**.

Název Názvy firem a jejích produktů vysazuji *kurzivou*. Kurzivou vysazuji také názvy kapitol, podkapitol a oddílů, na které se v textu odkazuji.

Odkaz Celá kniha je prošpikovaná křížovými odkazy na související pasáže. Není-li odkazovaný objekt (kapitola, obrázek, výpis programu, …) na některé ze sousedních stránek, je pro čtenáře tištěné verze doplněn o číslo stránky, na níž se nachází. Čtenářům elektronické verze stačí, když na něj klepnou, a použitý prohlížeč by je měl na odkazovaný objekt ihned přenést.

Citace Texty, které si můžete přečíst na displeji, např. názvy polí v dialogových oknech či názvy příkazů v nabídkách, vysazuji **tučným bezpatkovým písmem**.

Adresy Názvy souborů a internetové adresy vysazuji obyčejným bezpatkovým písmem.

`Program` Identifikátory a další části programů zmíněné v běžném textu vysazuji `neproporcionálním písmem`, které je v elektronických verzích pro zvýraznění tmavě červené.

`metoda(?)` Při odkazech na metody budu v závorkách za názvem metody vždy uvádět seznam typů jejich parametrů – např. `equals(Object)`. Nebude-li v danou chvíli jasné, jaké má zmiňovaná metoda parametry, budu do závorek psát otazník – např. `metoda(?)`.

`zadání` Ve výpisech komunikace s prostředím *JShell* bude zadání uživatele vysazeno tučně (v elektronických verzích pro zvýraznění tmavě červeně), a odpovědi prostředí netučně (a černě).

`Komentář` Pokud se v programech objeví komentáře, budou podbarveny zeleně.

Kromě výše zmíněných částí textu, které považuji za důležité zvýraznit nebo alespoň odlišit od okolního textu, najdete v textu ještě řadu doplňujících poznámek a vysvětlivek. Všechny budou v jednotném rámečku, který bude označen ikonou charakterizující druh informace, kterou vám chce poznámka či vysvětlivka předat.

Symbol jing-jang bude uvozovat poznámky, s nimiž se setkáte na počátku každé kapitoly. Zde vám vždy prozradím, co se v dané kapitole naučíte.

Píšící ruka označuje obyčejnou poznámku, která pouze doplňuje informace z hlavního proudu výkladu o nějakou zajímavost.

Symbol znamení raka označuje upozornění, že následný výklad se týká funkcionality, která je prozatím definována jako předběžná (preliminary). Lze očekávat, že bude v některé z příštích verzí zprovozněna, ale výsledné vlastnosti se mohou od stávajících lišit.

Otevřená schránka s dopisy označuje informace o projektu, s nímž budeme v dalším textu pracovat, nebo v něm najdete vzorové řešení aplikující probranou látku. Příslušný projekt získáte pomocí generátoru projektů popsaného výše.

Obrázek knihy označuje poznámku týkající se používané terminologie. Tato poznámka většinou upozorňuje na další používané termíny označující stejnou skutečnost nebo na konvence, které se k probírané problematice vztahují. Seznam všech terminologických poznámek najdete v rejstříku pod heslem „terminologie“.

Ruka s hrozícím prstem upozorňuje na věci, které byste měli určitě vědět a na které byste si měli dát pozor, protože jejich zanedbání vás většinou dostane do problémů.

Usměváček vás bude upozorňovat na různé tipy, kterými můžete vylepšit svůj program nebo zefektivnit svoji práci.

Mračoun vás naopak bude upozorňovat na různá úskalí programovacího jazyka nebo programů, s nimiž budeme pracovat, a bude vám radit, jak se těmto nástrahám vyhnout či jak to zařídit, aby vám alespoň pokud možno nevadily.

Brýle označují tzv. „poznámky pro šťouraly“, ve kterých se vás snažím seznámit s některými zajímavými vlastnostmi probírané konstrukce nebo upozorňuji na některé souvislosti, avšak které nejsou k pochopení látky nezbytné.

Odbočka – podšeděný blok

Občas je potřeba vysvětlit něco, co nezapadá přímo do okolního textu. V takových případech používám podšeděný blok se silnou čarou po straně. Tento podšeděný blok je takovou drobnou odbočkou od ostatního výkladu. Nadpis podšeděného bloku pak najdete i v podrobném obsahu mezi nečíslovanými nadpisy.

Zpětná vazba

Přes veškeré úsilí, které jsme knize já i moji spolupracovníci věnovali, nemohu vyloučit, že v textu či doprovodných příkladech zůstaly skryté nějaké chyby. Předem se za ně omlouvám. Objevíte-li proto v knize nějakou chybu nebo budete-li mít návrh na nějaké její vylepšení, pošlete prosím e-mail s předmětem 62_Java14_DOTAZ na adresu rudolf@pecinovsky.cz. Na stránku knihy http://knihy.pecinovsky.cz/62_j14lang se pak pokusím co nejdříve zanést příslušná errata s opravou, kterou zapracujeme do případného dalšího vydání.

Tento mail pošlete i v případě, když vám bude někde připadat text nepříliš srozumitelný nebo budete-li mít nějaký dotaz, ať už k vykládané látce či použitému vývojovému prostředí. Bude-li se dotaz týkat něčeho obecnějšího, zveřejním na stránce knihy odpověď i pro ostatní, které by mohl obdobný dotaz napadnout za pár dní, anebo jsou natolik ostýchaví, že si netroufnou sami se zeptat.

Dopředu se omlouvám, že vzhledem k velkému pracovnímu zatížení občas odpovídám na maily až se značným zpožděním.

Část I: Neobjektové konstrukce

Tato část vás nejprve seznámí s prostředím *JShell*, které budu ve svém příkladu využívat k tomu, abychom získali okamžitou odpověď na zadané programové obraty. Budeme tak moci od začátku zkoušet probírané konstrukce, aniž bychom museli používat něco, co jsme ještě neprobrali. Poté postupně probereme algoritmické konstrukce, které vyšší programovací jazyky nabízely ještě před příchodem objektového paradigmatu, kterému bude věnována druhá část.