

knihovna programátora

- Seznamuje s poslední verzí nejoblíbenějšího testovacího frameworku na platformě Java
- Na příkladech probírá základy i pokročilá témata
- Slouží jako učebnice, příručka i referenční manuál
- Uvádí různé příklady využití frameworku JUnit
- Obsahuje stručný úvod do teorie testování
- Doplněna o komplexní příklad testování všech vrstev aplikace

ARNOŠT HAVELKA
RUDOLF PECINOVSKÝ

JUnit 5

Jednotkové testování na platformě Java





knihovna programátora

ARNOŠT HAVELKA
RUDOLF PECINOVSKÝ

JUnit 5

Jednotkové testování na platformě Java

GRADA
Publishing

Upozornění pro čtenáře a uživatele této knihy

Všechna práva vyhrazena. Žádná část této tištěné či elektronické knihy nesmí být reprodukována a šířena v papírové, elektronické či jiné podobě bez předchozího písemného souhlasu nakladatele.

Neoprávněné užití této knihy bude **trestně stíháno**.

Arnošt Havelka a Rudolf Pecinovský

JUnit 5

Jednotkové testování na platformě Java

Vydala Grada Publishing, a.s.

U Průhonu 22, Praha 7

obchod@grada.cz, www.grada.cz

tel.: +420 234 264 401, fax: +420 234 264 400

jako svou 7022. publikaci

Odpovědný redaktor: Petr Somogyi

Návrh vnitřního layoutu: Rudolf Pecinovský

Zlom: Rudolf Pecinovský

Počet stran 296

První vydání, Praha 2018

Vytiskly Tiskárny Havlíčkův Brod, a. s.

© Grada Publishing, a.s., 2018

Cover Design © Grada Publishing, a. s., 2018

Cover Photo © Depositphotos

Názvy produktů, firem apod. použité v knize mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

ISBN 978-80-247-1496-7 (ePub)

ISBN 978-80-247-1225-3 (pdf)

ISBN 978-80-271-0733-9 (print)

Stručný obsah

Předmluvy autorů	22
Úvod	24
Část I: Teorie testování	31
1 Základy teorie testování	32
2 Metodika TDD	38
3 Jednotkové testy	42
Část II: Základy práce s frameworkem JUnit	55
4 Úvod do frameworku JUnit	56
5 Třída Assertions – potvrzení očekávání	70
6 Základní užitečné anotace	87
7 Deaktivace testů	103
8 Zvláštní druhy testů	115
9 Zvláštní druhy kontejnerů	139
Část III: Architektura a další pokročilá témata	151
10 Architektura frameworku JUnit	152
11 Rozšíření frameworku JUnit	165
12 Pokročilé funkce	181
Část IV: Nadstavby a rozšíření	203
13 Rozšířené testování s pomocí externích frameworků	205
14 Pokrytí kódu testy	225
15 Využití frameworku JUnit v projektech Spring	230
16 Nástroj Cucumber	261
Část V: Přílohy	271
A Rozdíly oproti předchozím verzím	272
B Potřebné vybavení a programy pro třetí a čtvrtou část	281
Použité informační zdroje	283
Rejstřík	288

Podrobný obsah

Předmluvy autorů	22
Rudolf Pecinovský	22
Doprovodné programy a jejich prohlížení	22
Arnošt Havelka	23
Úvod	24
Účel této publikace	24
Trocha historie	24
Jazyk Java versus platforma Java	25
Terminologie	26
Potřebné vybavení a doprovodné programy	26
Použité typografické konvence	27
Odbočka – podšeděný blok	29
Zpětná vazba	29
Část I: Teorie testování	31
1 Základy teorie testování	32
1.1 Význam a účel testování	32
1.2 Klasifikace testů	33
Metody testování	33
Strategie testování	33
Typy testů	34
Způsob testování	36
1.3 Náklady chyb	36
2 Metodika TDD	38
2.1 Motivace	38
2.2 Koncept	39
2.3 Příklad	40
3 Jednotkové testy	42
3.1 Charakteristika	42
3.2 Základní požadavky	43
3.3 Omezení	44
3.4 Pokrytí kódu testy	45
Princip	46
Měření pokrytí	47
3.5 Doporučení pro tvorbu testů	47
Obecné zásady	47
Co dodržovat	48
Co nedělat	49
Co netestovat	50
Doplněk k předchozím doporučením	50
3.6 Návrhové vzory, antivzory a pachy	51
3.7 Zástupné objekty a jejich použití	51

Typy používaných zástupných objektů	52
Atrapa (dummy object)	52
Testovací pahyl (test stub)	52
Testovací špión (test spy)	52
Mock-objekt (mock, mock-up, mock object)	52
Padělek (fake object, fake)	52
Vytváření zástupných objektů	53
Důvody pro používání zástupných objektů	53

Část II: Základy práce s frameworkem JUnit **55**

4 Úvod do frameworku JUnit	56
4.1 Požadovaná verze JDK	56
4.2 Sjednocení testovacího prostředí	56
4.3 Vytváříme první jednotkový test	60
4.4 Analýza výsledku testu	63
Kontejnery	63
Barvy	64
4.5 Vytváříme první skutečný test	64
4.6 Vyhodnocení souhrnné zprávy o průběhu testu	66
4.7 Testy pro starší verze frameworku	68
5 Třída Assertions – potvrzení očekávání	70
5.1 Představení	70
Doplnění chybových zpráv	71
Výhoda funkčního objektu	71
5.2 Výjimka <code>AssertionFailedError</code>	72
5.3 Metody pracující s jedním objektem	72
Sada metod <code>assertEquals</code>	73
Zvláštnosti typů <code>double</code> a <code>float</code>	73
Sady metod <code>assertFalse</code> a <code>assertTrue</code>	74
Sada metod <code>assertSame</code>	74
Sady metod <code>assertNotEquals</code> a <code>assertNotSame</code>	75
Sady metod <code>assertNull</code> a <code>assertNotNull</code>	76
5.4 Metody pracující s poli a iterátory	76
Sada metod <code>assertArrayEquals</code>	76
Sada metod <code>assertIterableEquals</code>	78
5.5 Funkční interfejs <code>Executable</code>	78
5.6 Sada metod <code>assertAll</code>	79
5.7 Metody řešící vyhazování výjimek	81
Funkční interfejs <code>ThrowingSupplier<T></code>	81
Sada metod <code>assertDoesNotThrow</code>	82
Sada metod <code>assertThrows</code>	82
5.8 Metody hlídající dobu zpracování akce	83
5.9 Metoda <code>assertLinesMatch</code>	84
5.10 Metody pro předčasné ukončení testu	85
6 Základní užitečné anotace	87
6.1 Životní cyklus testů	87
6.2 Testovací přípravek a organizační metody	88
6.3 Organizační metody v hierarchii dědění	90
Analýza zprávy o průběhu testu	92
6.4 @DisplayName – srozumitelnější pojmenování	95
6.5 TestInfo – informace o probíhajícím testu	96
6.6 TestReporter – dodání informací do souhrnu	97

6.7 Ukázka použití probraných rozšíření.....	97
6.8 Anotace @Tag a @Tags.....	101
Opakovatelnost.....	102
7 Deaktivace testů	103
7.1 Deaktivace testu.....	103
7.2 Podmíněné spuštění	105
@DisabledOnJre, @EnabledOnJre	106
@DisabledOnOs, @EnabledOnOs	106
@DisableIfEnvironmentVariable, @EnableIfEnvironmentVariable	107
@DisableIfSystemProperty, @EnableIfSystemProperty.....	107
@DisableIf, @EnableIf	108
Souhrnná zpráva	111
7.3 Třída Assumptions – přerušení nekorektního testu.....	111
Metody assumeFalse(?) a assumeTrue(?)	113
Metody assumingThat(?)	114
8 Zvláštní druhy testů	115
8.1 Opakované testy	115
Interfejs RepetitionInfo.....	116
Použití opakovaného testu.....	116
Úprava názvů instancí prostřednictvím anotace @RepeatedTest	117
8.2 Testy s parametry	121
Základy použití parametrického testu	121
Implicitní převody typů	123
Zadání jednoho argumentu	124
Anotace @ValueSource	124
Anotace @EnumSource	125
Anotace @MethodSource	127
Zadávání více argumentů	129
Anotace @CsvSource	131
Názvy testů s parametry.....	133
Anotace @CsvFileSource	134
Interfejs Arguments a anotace @MethodSource	135
Přítomnost dalších parametrů	136
9 Zvláštní druhy kontejnerů.....	139
9.1 Využití interfejsů	139
9.2 Vnitřní testovací třídy	142
9.3 Závěrečný vzorový příklad	143
Metoda setCountToZero(TestInfo)	144
Metoda print(int, TestInfo, boolean)	144
Analýza závěrečné souhrnné zprávy.....	148

Část III: Architektura a další pokročilá témata 151

10 Architektura frameworku JUnit	152
10.1 Základy architektury	152
Specifikace požadavků na framework JUnit.....	152
Základní oblasti funkcionality frameworku JUnit.....	153
Základní moduly frameworku JUnit.....	154
Projekty frameworku JUnit.....	155
Rozvoj API.....	156
10.2 Koncepte spouštění testů	158
Launcher	159
TestEngine	161
10.3 Izolace testů.....	163

11	Rozšíření frameworku JUnit	165
11.1	Úvod k rozšíření	165
	Klíčové vlastnosti rozšíření	165
	Základní principy při rozšiřování	166
11.2	Koncept rozšíření JU5	167
	Body rozšíření	167
	Funkční interfejsy	168
	Pořadí zpracování bodů rozšíření	169
	Registrace bodu rozšíření	169
	Kontext rozšíření	170
11.3	Způsoby registrace rozšíření	170
	Deklarativní registrace rozšíření	170
	Programová registrace rozšíření	172
	Automatická registrace rozšíření	175
	Chování rozšíření	176
11.4	Ukázka vlastního rozšíření	176
	Implementace rozšíření StatExtention	176
	Implementace anotace @Stat	179
	Implementace testu StatTests	179
12	Pokročilé funkce	181
12.1	Zpracování vstupních parametrů testovací metody	181
	Implementace rozšíření HelloResolver a anotace @Hello	182
	Použití rozšíření HelloResolver a anotace @Hello	183
	Limity zpracování parametrů	183
12.2	Ošetřování výjimek	184
	Implementace rozšíření IllegalArgumentExceptionHandler	184
	Použití rozšíření IllegalArgumentExceptionHandler	185
12.3	Testy s parametry (rozšířená témata)	185
	Předání jednoho argumentu pomocí anotace @ArgumentSource	186
	Implementace komponenty s interfejsem ArgumentProvider	186
	Test s anotací @ArgumentSource	186
	Konverze vstupních hodnot pro testy s parametry	187
	Implicitní konverze	187
	Explicitní konverze	188
	Předání více argumentů	189
	Interfejs ArgumentsAccessor	189
	Interfejs ArgumentsAggregator	189
	Vytvoření vlastní anotace @CsvToDrink	190
	Limity testů s parametry	191
12.4	Filtrování testů (doplňk k anotaci @Tag)	191
	Filtrování pomocí interfejsu Launcher	192
	Interfejs Launcher	192
	Příklad použití	192
	Filtrování testů nástrojem Maven	194
12.5	Dynamické testy	195
	Deterministicky generované testy	196
	Nedeterministicky generované testy	197
	Omezení dynamických testů	199
12.6	Testovací šablony	199

Část IV: Nadstavby a rozšíření 203

Souhrnné poznámky ke čtvrté části	204
Požadované znalosti	204
Rozdělení rozšiřujících nástrojů do kategorií	204

13	Rozšířené testování s pomocí externích frameworků	205
13.1	Tvorba potvrzení	205
	Framework <i>AssertJ</i>	206
	Konfigurace frameworku <i>AssertJ</i> pomocí nástroje <i>Maven</i>	206
	Základní použití	207
	Framework <i>Hamcrest</i>	207
	Konfigurace frameworku <i>Hamcrest</i> pomocí nástroje <i>Maven</i>	208
	Základní použití	208
13.2	Mockování	208
	Úvod do mockování	209
	Mock vs. stub	210
	Mock vs. spy	211
	Framework <i>Mockito</i>	212
	Konfigurace frameworku <i>Mockito</i> pomocí frameworku <i>Maven</i>	212
	Základní použití	212
	Alternativní použití	214
	Další možnosti mockování	214
	Limity použití	216
	Framework <i>EasyMock</i>	216
	Konfigurace frameworku <i>EasyMock</i> pomocí nástroje <i>Maven</i>	217
	Základní použití	217
	Alternativní použití	218
	Další možnosti mockování	218
	Limity použití	218
13.3	Testování speciálních případů	219
	Privátní konstruktor	219
	Framework <i>PowerMock</i>	220
	Framework <i>PowerMock</i>	220
	Konfigurace frameworku <i>PowerMock</i> pomocí nástroje <i>Maven</i>	220
	Testování statické metody	221
	Testování finální třídy	223
14	Pokrytí kódu testy	225
14.1	Nástroje pro výpočet pokrytí kódu testy	225
14.2	Metody měření pokrytí kódu	225
14.3	Ukázky nedostatečného pokrytí kódu	226
	Ukázka 1 (kalkulátor)	226
	Implementace	226
	Tvorba UT	227
	Ukázka 2 (metoda pro výpočet bonity klienta)	228
	Implementace	228
	Tvorba UT	228
14.4	Testování pomocí mutací	228
	Měření pokrytí	229
	Nástroj <i>PIT</i>	229
15	Využití frameworku <i>JUnit</i> v projektech <i>Spring</i>	230
15.1	Framework <i>Spring</i>	230
	Úvod do frameworku <i>Spring</i>	230
	<i>Spring framework</i>	231
	Konfigurace testů ve frameworku <i>Spring</i> pomocí nástroje <i>Maven</i>	231
	Implementace <i>HelloService</i>	232
	Tvorba testu pro <i>Spring Bean</i> pomocí anotace <i>@SpringJUnit4Config</i>	232
	Tvorba <i>Spring</i> testu pomocí standardního rozšíření <i>JU5</i>	233
15.2	Framework <i>Spring Boot</i>	234
	Konfigurace frameworku <i>Spring Boot</i> pomocí nástroje <i>Maven</i>	234
	Konfigurace aplikace	234
	Framework <i>Spring Boot</i>	235

Persistentní vrstva.....	237
Implementace.....	238
Tvorba testu	242
Servisní vrstva	244
Implementace.....	244
Tvorba testu	245
Prezentační vrstva	246
Implementace.....	246
Tvorba testu	247
15.3 Generování dokumentace REST služeb	249
Nástroj <i>Spring REST Docs</i>	249
Framework <i>Swagger</i>	250
Konfigurace Spring REST Docs pomocí nástroje <i>Maven</i>	251
Rozšíření testů pro přípravu vstupních dat dokumentace	251
Ukázka	255
Nástroj <i>AsciiDoctor</i>	257
Konfigurace <i>AsciiDoctor</i> pomocí nástroje <i>Maven</i>	257
Tvorba dokumentace	258
Ukázka	260
16 Nástroj <i>Cucumber</i>	261
16.1 Úvod do <i>Cucumber</i>	261
16.2 Konfigurace nástroje <i>Cucumber</i> pomocí nástroje <i>Maven</i>	261
Nástroj <i>Cucumber</i>	262
16.3 Ukázky tvorby testů.....	263
Ukázka č. 1 (jednoduchý kalkulátor)	263
Implementace jednoduchého kalkulátoru	263
Scénář chování kalkulátoru.....	264
Test kalkulátoru.....	265
Spuštění testu	265
Ukázka č. 2 (rozšířený kalkulátor)	266
Implementace rozšířeného kalkulátoru	266
Scénář chování kalkulátoru.....	268
Test kalkulátoru.....	269
Spuštění testu	269
Část V: Přílohy	271
A Rozdíly oproti předchozím verzím	272
A.1 Ještě jednou trocha historie	272
A.2 <i>JUnit</i> 4.12 – základní nástroje	273
Datové typy probírané ve druhé části	273
Pravidla pro klíčové metody	273
Získání názvu aktuálního testu	273
Třídy <i>Assert</i> a <i>Assume</i>	274
A.3 <i>JUnit</i> 3.8.2 – základní nástroje	275
A.4 <i>JUnit</i> 4.12 – pokročilé nástroje	276
A.5 Architektura verze <i>JUnit</i> 4.12.....	276
Rozšíření pomocí třídy <i>Runner</i>	276
<i>Spring framework</i>	278
Rozšíření prostřednictvím pravidel.....	278
Čtení jména testovací metody	279
B Potřebné vybavení a programy pro třetí a čtvrtou část.....	281
Použité informační zdroje.....	283
Rejstřík	288

Seznam programů

Výpis 4.1:	Definice třídy Main sloužící ke spuštění testovacích tříd	58
Výpis 4.2:	Prostoduchá definice testovací třídy TestHelloWorld_v5 a jednotkového testu helloWorld	62
Výpis 4.3:	Výpis zprávy o průběhu testu definovaného ve výpisu 4.2	62
Výpis 4.4:	Výpis zprávy o výsledku testu definovaného ve výpisu 4.2 v prostředí, které neumí zpracovat nastavení ANSI barev	64
Výpis 4.5:	Třída TestMinMax s ukázkami reálných testovacích metod	65
Výpis 4.6:	Souhrnná zpráva o průběhu testu definovaného ve výpisu 4.5	67
Výpis 4.7:	Definice testovací třídy TestHelloWorld_v4	68
Výpis 4.8:	Definice testovací třídy TestHelloWorld_v3	68
Výpis 4.9:	Výpis zprávy o průběhu testů při spuštění všech čtyř doposud definovaných testovacích tříd	69
Výpis 5.1:	Definice metody test_assertAll() ve třídě TestAssertAll	80
Výpis 5.2:	Zestručněná definice funkčního rozhraní ThrowingSupplier<T>	81
Výpis 6.1:	Definice testovací třídy TestSetUpTearDown	90
Výpis 6.2:	Výstup testů ve třídě TestSetUpTearDown a úvodní strom ze závěrečného souhrnu	91
Výpis 6.3:	Definice třídy TestAbstractLifeCycle	92
Výpis 6.4:	Definice třídy TestLifeCycle	93
Výpis 6.5:	Počátek zprávy o průběhu testů tříd LifeCycleTest a LifeCycle2Test	94
Výpis 6.6:	Definice testovací třídy TestParamDemo	98
Výpis 6.7:	Výstup testů TestParamDemo a úvodní strom ze závěrečného souhrnu	100
Výpis 6.8:	Definice testovací třídy TestTagTags	102
Výpis 7.1:	Třída TestDisable demonstrující možnosti nepodmíněného zakázání testu	104
Výpis 7.2:	Závěrečný souhrn po aktivaci tříd TestDisable a TestDisableClass	104
Výpis 7.3:	Třída TestDisableClass demonstrující možnosti nepodmíněného zakázání všech testů dané třídy	105
Výpis 7.4:	Ukázky použití podmíněného spuštění testu zadáním anotací @DisabledOnJre a @EnabledOnJre	106
Výpis 7.5:	Ukázky použití podmíněného spuštění testu zadáním anotací @DisabledOnOs a @EnabledOnOs	107
Výpis 7.6:	Ukázky použití podmíněného spuštění testu zadáním anotací @DisableIfEnvironmentVariable a @EnableIfEnvironmentVariable	108
Výpis 7.7:	Ukázky použití podmíněného spuštění testu zadáním anotací @DisabledIfSystemProperty a @EnabledIfSystemProperty	109
Výpis 7.8:	Ukázky použití podmíněného spuštění testu zadáním anotací @DisabledIf a @EnabledIf	111

Výpis 7.9:	Souhrnná zpráva o průběhu testů ve třídě TestEnableDisable.....	112
Výpis 7.10:	Ukázka použití metody assumeTrue	113
Výpis 7.11:	Ukázka použití metody assumingThat	114
Výpis 8.1:	Definice třídy TestRepetitions s ukázkami použití opakovaného testu	117
Výpis 8.2:	Souhrnná zpráva o průběhu testů třídy TestRepetitions.....	118
Výpis 8.3:	Definice třídy TestNamedRepetitions s ukázkami vlastního nastavení pro zobrazení názvu instance testu	119
Výpis 8.4:	Souhrnná zpráva o průběhu testů třídy TestNamedRepetitions	120
Výpis 8.5:	Třída TestSimpleParametrized s definicí jednoduchého parametrického testu	122
Výpis 8.6:	Strom ze závěrečné zprávy o testech ve třídě TestSimpleParametrized	122
Výpis 8.7:	Definice třídy TestDefaultTypeConversion.....	124
Výpis 8.8:	Úvod zprávy o průběhu testu třídy TestDefaultTypeConversion.....	124
Výpis 8.9:	Definice anotace @ValueSource	125
Výpis 8.10:	Definice třídy TestEnumSource.....	126
Výpis 8.11:	Závěrečná zpráva o testech ve třídě TestEnumSource.....	128
Výpis 8.12:	Definice třídy TestMethodSource s ukázkami testů s argumenty zadávanými prostřednictvím továrních metod	130
Výpis 8.13:	Strom ze závěrečné zprávy o průběhu testů z třídy TestMethodSource.....	131
Výpis 8.14:	Definice třídy TestCsvSource.....	132
Výpis 8.15:	Úvod závěrečné zprávy o testu třídy TestCsvSource	133
Výpis 8.16:	Definice třídy TestCsvFileSource demonstrující možnost zadání argumentů v souborech	135
Výpis 8.17:	Obsah souboru Root.txt.....	135
Výpis 8.18:	Počátek souhrnné zprávy o průběhu testu třídy TestCsvFileSource.....	135
Výpis 8.19:	Definice třídy TestMethodMultiSource demonstrující možnost zadání několika argumentů prostřednictvím metody	136
Výpis 8.20:	Počátek souhrnné zprávy o průběhu testu třídy TestMethodMultiSource.....	136
Výpis 8.21:	Definice třídy TestMultiParam demonstrující možnost použití informačních parametrů	137
Výpis 8.22:	Počátek závěrečné zprávy o testech třídy TestMultiParam	138
Výpis 9.1:	Definice interfejsu IBeforeAfter definujícího organizační metody a sadu pomocných metod pro zvětšování a zmenšování odsazení.	140
Výpis 9.2:	Definice testovací třídy TestIBeforeAfter.....	141
Výpis 9.3:	Úvod závěrečné zprávy o průběhu testů třídy TestIBeforeAfter.....	141
Výpis 9.4:	Zdrojový kód třídy TestInner demonstrující využití anotací @Nested a @Tag	144
Výpis 9.5:	Úvod závěrečné zprávy o průběhu testů třídy TestInner	148
Výpis 10.1:	Ukázka nastavení stabilní vlastnosti pro anotaci @DisplayName	157
Výpis 10.2:	Ukázka nastavení experimentální vlastnosti pro anotaci @CsvFileSource	157
Výpis 10.3:	Ukázka vlastního vyhledání testů (viz [28])	160
Výpis 10.4:	Ukázka vlastního spuštění a vyhodnocení testů (viz [28]).....	161
Výpis 10.5:	Ukázka izolace testů přes testovací třídu	164
Výpis 11.1:	Definice rozšíření pro interfejs BeforeEachCallback	171
Výpis 11.2:	Definice rozšíření pro interfejs AfterEachCallback	171
Výpis 11.3:	Ukázka testu pro deklarativní registraci rozšíření.....	171
Výpis 11.4:	Výstup běhu testovací třídy CustomTests	172
Výpis 11.5:	Definice rozšíření pro správu dočasného adresáře	173

Výpis 11.6:	Definice rozšíření pro tvorbu dočasných souborů	174
Výpis 11.7:	Ukázka testu pro programovou registraci rozšíření	175
Výpis 11.8:	Výstup běhu testovací třídy RegisterExtensionTest	175
Výpis 11.9:	Definice komplexního rozšíření pro zpracování statistik testů	177
Výpis 11.10:	Definice anotace @Stat	179
Výpis 11.11:	Ukázka testu využívajícího rozšíření pomocí anotace @Stat	180
Výpis 11.12:	Výpis z běhu testovací třídy StatTests	180
Výpis 12.1:	Definice anotace @Hello	182
Výpis 12.2:	Ukázka implementace třídy ParameterResolver	182
Výpis 12.3:	Ukázka použití třídy HelloResolver pro vlastní zpracování vstupních parametrů metod	183
Výpis 12.4:	Definice komponenty implementujícího interface TestExecutionExceptionHandler	185
Výpis 12.5:	Ukázka implementace třídy ParameterResolver	186
Výpis 12.6:	Ukázka testu s parametry pomocí implementace interface ArgumentsProvider	187
Výpis 12.7:	Ukázka testu s parametry pomocí implementace interface ArgumentsProvider	187
Výpis 12.8:	Ukázka použití implicitní konverze hodnot pro testy s parametry	188
Výpis 12.9:	Ukázka definice explicitního konvertoru pomocí interface ArgumentConverter	188
Výpis 12.10:	Ukázka použití explicitní konverze pomocí třídy CustomConverter	189
Výpis 12.11:	Ukázka použití standardního interface ArgumentsAccessor	189
Výpis 12.12:	Ukázka použití vlastní implementace interface ArgumentsAccessor	190
Výpis 12.13:	Ukázka vytvoření vlastní anotace @CsvToDrink pro více argumentů	191
Výpis 12.14:	Ukázka použití vlastní anotace @CsvToDrink pro více argumentů	191
Výpis 12.15:	Ukázka vyhledání testů pomocí interface Launcher (viz [28], kapitola 7.1)	193
Výpis 12.16:	Ukázka maven konfigurace pluginu maven-surefire-plugin	194
Výpis 12.17:	Ukázka použití deterministického generátoru testů	196
Výpis 12.18:	Ukázka použití nedeterministického generování testů	198
Výpis 12.19:	Definice anotace ParameterizedTest	200
Výpis 12.20:	Definice třídy ParameterizedTestExtension	201
Výpis 13.1:	Zapojení frameworku AssertJ do projektu	207
Výpis 13.2:	Ukázka použití frameworku AssertJ	207
Výpis 13.3:	Zapojení frameworku Hamcrest do projektu	208
Výpis 13.4:	Ukázka použití frameworku Hamcrest	209
Výpis 13.5:	Zapojení frameworku Mockito do projektu	212
Výpis 13.6:	Základní použití frameworku Mockito	213
Výpis 13.7:	Ukázka použití frameworku Mockito s anotací @Mock	214
Výpis 13.8:	Ukázka použití frameworku Mockito s anotací @Spy	215
Výpis 13.9:	Ukázka použití frameworku Mockito s anotací @Captor	216
Výpis 13.10:	Zapojení frameworku Mockito do projektu	217
Výpis 13.11:	Základní použití frameworku EasyMock	217
Výpis 13.12:	Ukázka použití frameworku EasyMock s anotací @Mock	218
Výpis 13.13:	Ukázka použití testu na privátní konstruktor třídy	219
Výpis 13.14:	Zapojení frameworku PowerMock do projektu	221
Výpis 13.15:	Definice třídy SequenceGenerator používající statickou inicializaci objektu	221

Výpis 13.16:	Definice finální třídy Person používající statickou metodu v konstruktoru objektu	222
Výpis 13.17:	Ukázka použití frameworku PowerMock pro statickou metodu	223
Výpis 13.18:	Definice konstruktoru třídy Person s využitím statické metody	223
Výpis 13.19:	Ukázka použití frameworku PowerMock pro finální třídu	224
Výpis 14.1:	Definice třídy Calculator s metodou pro součet vstupních hodnot	227
Výpis 14.2:	Definice testovací třídy CalculatorTest	227
Výpis 14.3:	Metoda rate pro výpočet bonity klienta	228
Výpis 14.4:	Definice testovací třídy BonityRatingTest	229
Výpis 15.1:	Základní konfigurace Maven projektu pro testy ve frameworku Spring	231
Výpis 15.2:	Definice Spring Bean ve třídě HelloService	232
Výpis 15.3:	Ukázka Spring testu pomocí anotace @SpringJUnitConfig	233
Výpis 15.4:	Ukázka Spring testu pomocí standardního rozšíření @ExtendWith	233
Výpis 15.5:	Základní konfigurace Maven projektu pro framework Spring Boot	235
Výpis 15.6:	Konfigurace Spring Boot aplikace	237
Výpis 15.7:	Definice entity City	239
Výpis 15.8:	Definice JPA repository ve třídě CityRepository	240
Výpis 15.9:	Zadání struktury a dat databáze v souboru db.changelog-master.yaml	241
Výpis 15.10:	Ukázka integračního testu pomocí technologie JDBC	243
Výpis 15.11:	Ukázka integračního testu na třídu CityRepository	243
Výpis 15.12:	Definice servisní třídy CityService	244
Výpis 15.13:	Ukázka integračního testu na třídu CityService	245
Výpis 15.14:	Definice REST služby ve třídě CityController	246
Výpis 15.15:	Ukázka testu ControllerBeanTest na třídu CityController	247
Výpis 15.16:	Ukázka testu ControllerRestTest na RESTful službu	248
Výpis 15.17:	Konfigurace Maven projektu pro nástroj Spring REST Docs	252
Výpis 15.18:	Ukázka testu ControllerRestDocsTest na RESTful službu – část 1	252
Výpis 15.19:	Ukázka testu ControllerRestDocsTest na RESTful službu	254
Výpis 15.20:	Ukázka vygenerovaného souboru http-request.adoc	255
Výpis 15.21:	Ukázka vygenerovaného souboru response-fields.adoc	256
Výpis 15.22:	Konfigurace Maven projektu pro nástroj AsciiDoctor	257
Výpis 15.23:	Předpis pro generování dokumentace v souboru api-guide.adoc	259
Výpis 16.1:	Maven závislosti pro nástroj Cucumber	263
Výpis 16.2:	Implementace jednoduchého kalkulátoru ve třídě SimpleCalculator	264
Výpis 16.3:	Zadání jednoduchého kalkulátoru v souboru CalcSimple.feature	265
Výpis 16.4:	Implementace testu ve třídě SimpleCalcStepdefs	266
Výpis 16.5:	Implementace JUnit testu pro Cucumber	266
Výpis 16.6:	Implementace kalkulátoru ve třídě AdvancedCalculator	267
Výpis 16.7:	Zadání rozšířeného kalkulátoru v souboru CalcAdvanced.feature	268
Výpis 16.8:	Implementace testu ve třídě AdvancedCalcStepdefs	270
Výpis A.1:	Ukázka řešení testu s parametry v JU4	277
Výpis A.2:	Ukázka testu na definovanou Spring Bean v JU4	278
Výpis A.3:	Ukázka čtení jména testovací metody v JU4 (pomocí pravidla @Rule)	280

Seznam obrázků

Obrázek 1.1: Diagram růstu nákladů na opravu nalezené chyby v čase	37
Obrázek 10.1: Prezentace výsledku běhu testovací třídy BasicTests v nástroji Eclipse	154
Obrázek 10.2: Schéma dekompozice JUnit 5 [40]	155
Obrázek 10.3: Schéma artefaktů JUnit 5 [41]	156
Obrázek 10.4: Schéma artefaktů JU5 a jejich vzájemné závislosti	157
Obrázek 11.1: Pořadí zpracování bodů rozšíření v JU5 (viz podkapitulu 5.11 v [28]).....	169
Obrázek 12.1: Seznam vytvořených a spuštěných testů z běhu deterministického generování testů.....	197
Obrázek 12.2: Seznam vytvořených a spuštěných testů z běhu nedeterministického generování testů.....	199
Obrázek 13.1: Schéma komponent pro komponentu Person	209
Obrázek 13.2: Schéma komunikace komponent s použitím mock-objektu	211
Obrázek 13.3: Schéma komunikace komponent s použitím stub-objektu.....	211
Obrázek 13.4: Schéma komunikace komponent s použitím spy objektu	212
Obrázek 14.1: Ukázka výstupu nástroje Cobertura v Jenkins [59].....	226
Obrázek 15.1: Generátor Spring Boot projektů	238
Obrázek 15.2: Přehled nástrojů pro dokumentaci RESTful služeb [58]	250
Obrázek 15.3: Ukázka připravených vstupních dat (tzv. snippet) v nástroji Eclipse	256
Obrázek 15.4: Ukázka vygenerované dokumentace pro REST API ve formátu HTML	260
Obrázek 16.1: Význam syntaxe Gherkin [10].....	262
Obrázek 16.2: Výsledek běhu testovací třídy RunCukestest.....	267
Obrázek 16.1: Hierarchie interfejsu Runner v JU4	277
Obrázek B.1: Schéma dekompozice zdrojových kódů (vzorových příkladů použitých v knize).....	281

Seznam tabulek

Tabulka 7.1: Předdefinované proměnné použitelné ve skriptech.....	110
Tabulka 8.1: Implicitní převody typů, převzato z příručky JUnit User Guide [6]	123
Tabulka 10.1: Seznam možných stavů v anotaci @API.....	158
Tabulka 11.1: Seznam dostupných interfejsů pro body rozšíření.....	168
Tabulka 13.1: Srovnání jednotlivých druhů syntaxí pro konstrukci potvrzení.....	206
Tabulka 13.2: Výhody a nevýhody použití mock a stub řešení	211
Tabulka 13.3: Seznam referencí frameworku PowerMock.....	220
Tabulka 15.1: Seznam běžně používaných metod Spring Rest Docs dokumentace	254
Tabulka A.1: Krátký přehled posledních revizí hlavních verzí frameworku JUnit.....	273
Tabulka A.2: Pseudokořenový balíček a přehled ekvivalentních datových typů probraných ve druhé části	274
Tabulka A.3: Pseudokořenový balíček a přehled názvů metod ve verzi 3 a odpovídajících anotací ve verzi 5.....	275
Tabulka A.4: Mapování pokročilých funkcí JUnit 4 a JUnit 5	276
Tabulka A.5: Seznam standardních pravidel JU4	279
Tabulka B.1: Seznam technologií použitých ve zdrojových kódech (pro vzorové příklady)	282

Seznam odboček – podšeděných bloků

Jazyk Java versus platforma Java	25
Odbočka – podšeděný blok	29
Framework <i>PowerMock</i>	220
Nástroj <i>PIT</i>	229
<i>Spring framework</i>	231
Framework <i>Spring Boot</i>	235
Framework <i>Swagger</i>	250
Nástroj <i>Cucumber</i>	262

Používané pojmy a zkratky

Zkrat- ka	Termín	Význam
API	Application Programming Interface (Programové rozhraní aplikace)	Rozhraní, přes které mohou jiné programy s danou aplikací komunikovat.
	Argument	Hodnota předávaná metodě v parametru a označovaná někdy jako <i>skutečný parametr</i> (viz Parametr).
	Black-box testing (Testování metodou černé skříňky)	Testování aplikace bez detailní znalosti její implementace. Testuje se pouze, zda testovaný objekt odpovídá na zadané podněty zadaným způsobem.
	Callback	Volané metodě (resp. oslovenému objektu) je v parametru předán objekt, jehož prostřednictvím pak tato metoda nebo její vlastník komunikuje s volajícím objektem.
DAO	Data Access Object (Objekt pro přístup k datům)	Objekt poskytující abstraktní rozhraní k nějakému typu databáze nebo jinému mechanismu trvalého uložení dat.
DB	Databáze	Sada dat uložená prostřednictvím nějakého systému pro řízení báze dat (viz SŘDB). V praxi se termín <i>databáze</i> často (byť ne zcela správně) používá v širším smyslu zahrnujícím jak uložená data, tak i software (SŘBD).
DTO	Data Transfer Object (Přeppravka)	Označení objektu sloužícího pro přenos dat mezi jednotlivými částmi aplikace (viz Přeppravka).
	Datovod	Instance některého z potomků interfejsu <code>java.util.stream.BaseStream</code>
DBMS	Database Management System	Viz SŘDB – Systém pro řízení báze dat.
	Fluent API	API umožňující konstrukci objektu či výrazu plynulou a přirozenou cestou.
JU4	<i>JUnit 4</i>	Čtvrtá verze frameworku <i>JUnit</i> .
JU5	<i>JUnit 5</i>	Pátá verze frameworku <i>JUnit</i> .

Zkrat- ka	Termín	Význam
IS	Informační systém (Information System)	Systém složený z hardwaru a softwaru sloužící nějakému účelu (např. ERP či CRM).
IDE	Integrated Development Environment (Integrované vývojové prostředí)	Program sdružující sadu nástrojů usnadňujících jednotlivé činnosti při vývoji programů (editace, překlad, ladění, refaktorigace, tvorba testů, sestavování, správa verzí, řízení projektu, ...).
	Interface (rozhraní)	Signatura plus kontrakt, které musí implementace daného rozhraní dodržet. Své rozhraní má každá entita programu: balíček, třída, metoda, proměnná, ...
	Interface type (interfejs)	Programová konstrukce definující v jazyku <i>Java</i> datový typ reprezentující rozhraní.
IoC	Inversion of Control (Inverze řízení)	Návrhový vzor řešící problematiku vzájemné závislosti frameworku a komponent, které jej využívají.
Java EE	Java Enterprise Edition	Nadstavba standardní edice platformy <i>Java</i> (<i>Java SE</i>) určená pro tvorbu distribuovaných aplikací (aplikací běžících na několika spolupracujících počítačích) používaná např. v informačních systémech podniků.
JPA	Java Persistence API	Standard pro tvorbu perzistentní vrstvy pomocí objektově-relačního mapování (viz ORM).
	JavaScript	Dynamický jazyk používaný pro tvorbu uživatelského rozhraní webových aplikací.
	Jednotkové testování (Unit testing)	Tvorba testů ověřujících funkcionality jednotek zdrojového kódu (metod, tříd, komponent, ...).
JVM	Java Virtual Machine	Virtuální stroj <i>Javy</i> .
	Mockování	Vytváření mock-objektů, tj. falešných instancí dané třídy s definovaným výstupem pro definované vstupy.
	Model rozšíření (Extension Model)	API pro rozšiřování funkcí a vlastností frameworku.
ORM	Object-relational mapping (Objektově relační mapování)	Technologie zajišťující automatickou konverzi dat mezi relačním modelem používaným v databázi a objektovým modelem používaným v programu.
	Parametr	Proměnná, jejímž prostřednictvím předává volající program volané metodě potřebná data – argumenty.
	Přepřevka (Crate, Transfer Object)	Návrhový vzor doporučující jak definovat objekt sloužící k přenosu skupin dat mezi částmi aplikace.
	Refactoring (Refaktorigace kódu)	Úprava zdrojového kódu nemění jeho funkcionality, ale zlepšující jeho architekturu.

Zkrat- ka	Termín	Význam
REST	Representational state transfer	Architektura rozhraní, navržená pro distribuované prostředí.
	Rozšíření (extension)	Možnost doplnění funkčnosti frameworku.
SLA	Service Level Agreement	Smlouva mezi poskytovatelem služby a uživatelem definující fungování požadované služby (dostupnost, výkon, počet uživatelů atd.).
	Sestavení aplikace	Poskládání přeložených částí a dat do výsledného tvaru, např. do souboru (nebo sady souborů) typu JAR.
SW	Software	Aplikace či IS zajišťující nějakou funkčnost.
SŘDB	Systém pro řízení báze dat (Database Management System)	Sada programů umožňující přístup k datům uloženým v databázi a efektivní manipulaci s těmito daty.
TDD	Test Driven Development (Vývoj řízený testy)	Metodika vývoje požadující definici testů před zahájením implementace. Vlastní vývoj pak spočívá v úpravách programu vedoucích k úspěšnému projití daných testů.
	Test Coverage (Pokrytí kódu testy)	Metrika udávající, jaké procento zdrojového kódu je pokryto existujícími testy.
	Testovací scénář	Testovací scénář představuje návod, podle kterého se testuje daná funkcionálnita.
UAT	Uživatelské akceptační testy (User Acceptance Test)	Testy používané při akceptaci díla, tj. při jeho přebírání zákazníkem, přičemž dílem může být jak nový program, tak pouze jeho dílčí změna.
UI	User Interface (Uživatelské rozhraní)	V kontextu této práce je jím myšleno webové rozhraní (HTML, <i>JavaScript</i> a CSS).
	Unit testing	Viz <i>Jednotkové testování</i> .
	White-box testování (Testování metodou bílé/skleněné skříňky)	Testování entity se znalostí její implementace.

Předmluvy autorů

Rudolf Pecinovský

Kniha vznikla rozpracováním diplomové práce Arnošta Havelky ([\[21\]](#)), která se mi natolik líbila, že jsem mu nabídl, že bychom ji mohli po jistém přepracování zkusit společně nabídnout k vydání nakladatelství *Grada*.

V průběhu práce se však ukázalo, že Arnošt si rozumí spíše se zkušenějšími programátory a výklad pro začínající uživatele není jeho oblíbená parketa. Tak jsme se dohodli, že si text rozdělíme. Teoretický úvod převezmeme z jeho práce s tím, že jej na některých místech trochu upravím, aby mu porozuměli i nepřiliš zkušení uživatelé. Druhou část určenou pro začínající uživatele napíšu já a další dvě části určené spíše pro ty zkušenější napíše on. Obdobně jsme si rozdělili i tvorbu přílohy pojednávající o testech vytvořených pro minulé verze frameworku a o možnostech jejich dalšího využívání. Tak vznikla kniha, kterou držíte v ruce.

Doprovodné programy a jejich prohlížení

Protože jsem text knihy i lámal, aby optimálně zaplnil stránky, měl bych pro vás ještě jednu informaci. 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, i když je občas bylo třeba kvůli tomu o stránku či dvě posunout. 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 rozбором obsahu daného výpisu a s případnými odkazy na jeho jednotlivé řádky.

Arnošt Havelka

Každá cesta někde začíná a končí, většinou však nebývá přímočará. Stejně to platí i pro knihu, kterou právě držíte v ruce. Její osud rozhodně nebyl přímočarý, místy musím přiznat i napínavý, zejména když se blížil termín uzávěrky. V každém případě si kniha našla cestu do vašich rukou, za což bych chtěl velmi poděkovat. Vážím si toho.

V této části samozřejmě nemůžu vynechat poděkování, které patří především mé rodině za nezměrnou trpělivost a podporu po celou dobu tvorby knihy. Velký dík samozřejmě patří také Rudolfovi, neboť bez jeho cenných rad a vedení by asi kniha nevznikla. Určitě by nevznikla v takové kvalitě, do jaké se nám ji společně podařilo dostat. Začátečnické nadšení a zápal je výborná věc, někdy však k úspěchu prostě nevede.

Závěrem svého úvodního slova bych rád ještě požádal o zpětnou vazbu, která je důležitá u každého díla (nejen u knihy). Pokud se vám kniha líbila, tak bych velmi ocenil udělení hvězdičky na adrese <https://github.com/arnosthavelka/junit-poc>, kde jsem připravoval všechny ukázkové příklady k frameworku *JUnit*. Pozitivní zpětná vazba mě může motivovat k další tvůrčí práci. Pokud máte jakékoliv konstruktivní připomínky či výhrady, tak neváhejte a vytvořte na výše uvedeném odkazu podnět (issue).

Úvod

Účel této publikace

Tato kniha by měla být rychlou úvodní příručkou soužící dvěma skupinám uživatelů. Její první a druhá část je koncipována tak, aby byla užitečná i pro programátory pracující na platformě Java, kteří prozatím nemají žádné zkušenosti s jednotkovým testováním. Přestože vám nepředvede všechny detaily jednotkového testování, pokusí se poskytnout vše, co potřebujete vědět, abyste se mohli ponořit do tvorby jednotkových testů vašich programů využívajících platformu *JUnit 5*, přesněji *JUnit 5.3.1*.

Současně jsme se snažili, aby tyto vstupní části přinesly potřebné informace i zkušenějším programátorům, kteří již s frameworkem *JUnit* pracují a potřebují se seznámit s novinkami, které přináší jeho pátá verze. Těm zkušenějším jsou navíc určeny i následující části. Třetí část probírá architekturu frameworku a seznamuje s jeho pokročilejšími funkcemi, čtvrtá část se zaměřuje na spolupráci frameworku s jinými aplikacemi a frameworky, které mohou jeho služeb využít a rozšířit jeho funkcionalitu.

Trocha historie

Framework *JUnit* představuje jedno z mnoha dostupných řešení pro tvorbu jednotkových testů. Je portem původního frameworku *SUnit*, který v roce 1989 vytvořil v jazyku *Smalltalk* a publikoval Kent Beck¹. V roce 1997 letěl s Erichem Gammou² z Zürichu na konferenci OOPSLA do Atlanty. V letadle se nudili, a tak převeďli Beckovu knihovnu na platformu *Java*, jejíž popularita tehdy rychle rostla a stávala se postupně hlavním programovacím jazykem vstupních kurzů programování. Svůj výtvar nazvali *JUnit*.

-
- ¹ Jeden z guru moderního programování. Mimo jiné iniciátor vzniku metodiky TDD jako odvozeniny zásady *nejprve testy* v jím vytvořené metodice vývoje nazvané *extrémní programování*.
 - ² Další z guru objektového programování. Mimo jiné spoluautor klíčové publikace [\[15\]](#), která výrazně ovlivnila moderní programování a stala se jednou z jeho biblí.

Jazyk *Java* versus platforma *Java*

Když v roce 1995 představovala firma *Sun* jazyk a platformu *Java*, tak si její marketingové oddělení neuvědomilo, že pod stejným názvem představuje dva naprosto rozdílné produkty.

Použitý programovací jazyk určuje způsob vyjadřování vašich myšlenek. Použitá platforma specifikuje prostředí, v němž program napsaný v daném programovacím jazyku běží.

Řada komentátorů si tento rozdíl neuvědomuje dodnes a občas tyto pojmy dokonce zaměňuje. Když např. *Microsoft* na přelomu století propagoval chystanou platformu *.NET*, uváděl, že platforma *Java* nabízí pouze jeden programovací jazyk, kdežto oni uvedou svoji platformu se čtyřmi jazyky, jejichž sada se vbrzku rozroste o desítku dalších. Jenomže už v té době bylo na platformě *Java* možno použít přes 80 programovacích jazyků a v dnešní době jich je již několik set. Mezi nejznámější patří (dle abecedy) Clojure, Cobol, Fortran, Groovy, JavaScript, Kotlin, Lisp, Pascal, Perl, PHP, Prolog, Python, R, Ruby, Scheme či Visual Basic.

V textu knihy se proto budeme snažit důsledně rozlišovat, kdy hovoříme o platformě *Java* a kdy o stejnojmenném programovacím jazyku.

Na konferenci svůj čerstvý výtvar představili. Popularita jejich frameworku raketově rostla a zanedlouho bylo nemyslitelné, že by nějaké vývojové prostředí nemělo zabudovanou jeho podporu. Popularita tohoto frameworku inspirovala vývojáře pracující na jiných platformách (PHP, JavaScript, *.NET*, Python, ...), aby framework převedli na svoji oblíbenou platformu, takže nyní se s jeho „sourozenci“ setkáte na všech jen trochu rozšířených platformách (wikipedie uvádí přes 80 portů).

Martin Fowler, známý počítačový guru, jednou charakterizoval význam tohoto frameworku parafrází známého Churchillova výroku o letecké bitvě o Anglii, když řekl: „*Při vývoji softwaru ještě nikdy nevděčilo tolik lidí tak moc tak málo řádkům kódu.*“³

První verze frameworku byly totiž opravdu malé. Jenomže každá další verze chce uživatelům umožnit, aby mohly spouštět i testy napsané pro minulou verzi, takže pak obsahuje minulou i novou verzi frameworku, takže framework pomalu bobtná.

Jak už jsme naznačili, framework se postupně vylepšoval. V současné době je nejrozšířenější verze 4, která se narodila v roce 2004 a do svého konceptu zapracovala generické datové typy a anotace zavedené v tehdy nové *Javě* 5.

³ V originále „*Never in the field of software development have so many owed so much to so few lines of code*“

Další výrazná změna přišla s verzí *JUnit 5*, jež se narodila v roce 2017 záhy po uvedení *Javy 9*, která zavedla moduly. Verze 5 se pomalu prosazuje, ale řada programátorů dává ze setrvačnosti přednost verzi předchozí (po těch letech si na ni už zvykli). Tato kniha by vás chtěla s novou verzí seznámit a ukázat vám její výhody a možnosti jejího použití v nejrůznějších vývojových nástrojích.

Terminologie

Jak už bylo řečeno, různé části knihy se snaží vyjít vstříc různým skupinám uživatelů. Tomu odpovídá i použitá terminologie.

V první a druhé části preferujeme standardní českou terminologii, protože je pro začátečníky srozumitelnější. Navíc zkušenost ukazuje, že si pak význam jednotlivých termínů dokáží snadněji zapamatovat.

Ve třetí a čtvrté části respektujeme, že zkušenější programátoři se většinou pohybují v prostředí zahlceném slangovými výrazy nebo dokonce v prostředí, kde spolu jednotliví spolupracovníci hovoří anglicky. Tito programátoři již často českou terminologii nepoužívají. Proto v těchto částech částečně rezignujeme na „správnou češtinu“ a dáváme přednost slangovým výrazům odvozeným z angličtiny.

Potřebné vybavení a doprovodné programy

Pro úspěšné studium této knihy potřebujete mít instalované vývojové prostředí, které vám umožní vyvíjet program nad platformou *Java 8* nebo novější. Jak už jsme naznačili na začátku úvodu, nemusí to být nutně programy v jazyce *Java*, ale měly by to být programy běžící nad platformou *Java*.

Druhou věcí, kterou budete potřebovat, je framework *JUnit* verze 5, resp. některá z jeho následujících podverzí (v době vydání knihy to byla verze 5.3.0). Používáte-li některé z rozšířených profesionálních integrovaných vývojových prostředí (IDE), je velmi pravděpodobné, že už v něm bude tato verze frameworku zabudována.

K tomu, abyste mohli popisovaný framework plnohodnotně využívat, je vhodné se občas začít do jeho programátorské dokumentace. Tu najdete na adrese <https://junit.org/junit5/docs/current/api>. O její off-line verzi použitelné i v případě, kdy nejste aktuálně připojeni k internetu, prozatím nevíme.

Další věcí, která by se vám mohla hodit, jsou doprovodné programy. Mezi nimi najdete všechny příklady uvedené v knize a navíc i několik dalších. Způsob jejich získání záleží na tom, ke které části knihy je potřebujete:

- V první části žádné doprovodné programy nejsou.
- Doprovodné programy k výkladu v druhé části určené pro programátory začínající s frameworkem *JUnit 5* najdete na webové stránce knihy na adrese http://knihy.pecinovsky.cz/54_junit5.

Kromě doprovodných programů najdete na této stránce časem i výklad pasáží, které se již do knihy nevešly⁴, např. výklad některých návrhových vzorů. Bude-li potřeba, objeví se zde i errata a odpovědi na vaše případné dotazy.

- Popis získání doprovodných programů k výkladu ve třetí a čtvrté části určené pro zkušenější uživatele frameworku *JUnit 5* je trochu složitější, a proto jsme jej umístili do přílohy [Potřebné vybavení a programy pro třetí a čtvrtou část](#) na straně [281](#).

Použité typografické konvence

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

Termíny První výskyt nějakého termínu a další texty, které chceme zvýraznit, jsou vysazeny **tučně**.

Název Názvy firem a jejich produktů jsou vysazeny *kurzivou*. Kurzivou jsou vysazeny také názvy kapitol, podkapitol a oddílů, na které se v textu odkazujeme.

[Odkaz](#) Celá kniha je prošpikovaná křížovými odkazy na související pasáže. Není-li odkazovaný objekt (kapitola, obrázek, výpis programu, ...) na stejné stránce nebo 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, jsou vysazeny **tučným bezpatkovým písmem**.

Adresy Názvy souborů a internetové adresy jsou vysazeny obyčejným bezpatkovým písmem.

⁴ S nakladatelem jsme se na počátku dohodli na cca 300 stránkách a do nich jsme se museli vejít

- Program** Identifikátory a další části programů zmíněné v běžném textu jsou vysazeny **neproporcionálním písmem**, které je v elektronických verzích pro zvýraznění tmavě červené.
- metoda(?)** Při odkazech na metody je v závorkách za názvem metody vždy uváděn seznam typů jejich parametrů – např. `equals(Object)`. Nebude-li v danou chvíli jasné, jaké má zmiňovaná metoda parametry, nebo se bude jednat o odkaz na celou skupinu přetížených metod, bude v závorkách otazník – např. `metoda(?)`.
- zadání** Ve výpisech programů a odpovědí počítače budou části, které chceme zdůraznit, vysazeny tučně (v elektronických verzích pro zvýraznění tmavě červeně) a budou navíc i trochu jinak podbarveny.

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



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



Otevřená schránka s dopisy označuje informace o projektu, s nímž budeme v dalším textu pracovat, nebo v něm najdete vzorové řešení aplikující probíranou látku.



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.



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



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



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.



Mračoun upozorňuje na různá úskalí a radí, jak se těmto nástrahám vyhnout či jak to zařídit, aby vám alespoň pokud možno nevadily.

Odbočka – podšeděný blok

Občas je potřeba vysvětlit něco, co nezapadá přímo do okolního textu. V takových případech používáme 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 textu oba věnovali, nemůžeme vyloučit, že v textu či doprovodných příkladech zůstaly skryté chyby. Předem se za ně omlouváme. Objevíte-li v knize nějakou chybu nebo budete mít návrh na nějaké její vylepšení, neostýchejte se napsat na adresu rudolf@pecinovsky.cz (pro první dvě části) nebo adresu arnost.havelka@gmail.com. (pro první, třetí a čtvrtou část). Pokusíme se co nejdříve zanést na stránku knihy příslušná errata s opravou, kterou pak zapracujeme do případného dalšího vydání.

Bude-li vám někde připadat text nepřiliš srozumitelný nebo budete mít nějaký dotaz, ať už k vykládané látce či použitému frameworku, pošlete [mail s předmětem DOTAZ JUNIT 5](mailto:DOTAZ_JUNIT@pecinovsky.cz). Bude-li se dotaz týkat něčeho obecnějšího, zveřejníme na stránce knihy http://knihy.pecinovsky.cz/49_adv_txt odpověď i pro ty ostatní, které by mohl obdobný dotaz napadnout za pár dní, anebo jsou natolik ostýchaví, že se netroufnou sami zeptat.