LEKCE 4

Testovací třída

Co se v lekci naučíme

V této lekci vytvoříme svoji první třídu – testovací. Vysvětlíme si, co je to testovací přípravek, a naučíme se jej definovat. Poté definujeme své první testy. Na závěr si vysvětlíme, jaký je rozdíl mezi zprávou a metodou.

Projekt:
V této lekci pokračujeme v používání projektu `101a_01_Tvary`.

54. Posledně jsi říkal, že vše, co si budu chtít zapamatovat, mohu před restartem virtuálního stroje uložit na disk. Mohu si nějak uložit i ty svoje experimenty?

Můžeš – např. tak, že definuješ třídu, kterou naučíš zareagovat na nějakou zprávu tak, že tvoji činnost zopakuje. *BlueJ* umí sledovat tvoji činnost a vytvořit program, který ji na požádání zopakuje. Pracuje přitom podobně jako záznamníky maker v některých kancelářských programech.

55. To bych si ráda vyzkoušela.

Proč ne. Jenom bych tě chtěl upozornit, že tento mechanizmus *BlueJ* poskytuje pouze pro testovací třídy. Ostatní druhy tříd musíš programovat klasicky, tj. napsat jejich program jako správně formulovaný text. Než ale začneme programy opravdu psát sami, využijeme této schopnosti *BlueJ* a necháme jej psát programy za nás.

56. Proč má zrovna nějaká testovací třída takové výjimečné postavení?

Já bych řekl, že je to proto, že koncepce testů použitá v knihovně *JUnit*, kterou *BlueJ* využívá, má několik vlastností, které naprogramování takovéhoto způsobu přípravy testů usnadňují. Navíc testy velmi často sestávají pouze z jednoduché posloupnosti volání zpráv, které pracují s předem známou sadou dat. Autoři *BlueJ* proto těchto výhod využili a studenti nyní mohou vytvářet jednoduché programy, i když ještě vůbec neumí programovat.

Testovací přípravek

57. Tak mi ukaž, jak mohu takovou třídu vytvořit.

Nejprve ti musím povědět něco o testovacích třídách obecně, aby ti pak bylo jasné, proč budeme dělat věci právě tak, jak ti budu ukazovat.

Jak jsi jistě z názvu odhadla, testovací třídy slouží k testování programů. K otestování chování nějaké části programu většinou nestačí jediný test. Velmi často je ale vhodné, aby celá skupina testů začínala ze stejného výchozího stavu. Testovací třídu můžeš chápat jako objekt schopný realizovat skupinu testů se společným výchozím stavem.

Výchozí stav, jenž je pro tuto skupinu testů společný, označujeme jako *testovací přípravek*. Požádáme-li testovací třídu o spuštění libovolného připraveného testu, vytvoří nejprve testovací přípravek a ten pak testuje.

Nejprve proto musíme testovací třídě ukázat, jak má připravit testovací přípravek, a pak můžeme s tímto přípravkem provádět různé experimenty, které budeme vydávat za testy.

Vytvoření nové třídy

58. Takže jestli jsem to dobře pochopila, začneme tím, že vytvoříme testovací přípravek.

Ne tak docela. Než začneme vytvářet přípravek, musíme mít nejprve vytvořenou testovací třídu. Začneme tedy vytvořením této třídy.

59. Nechytej mě za slova – to je přece jasné. Tak se předveď!

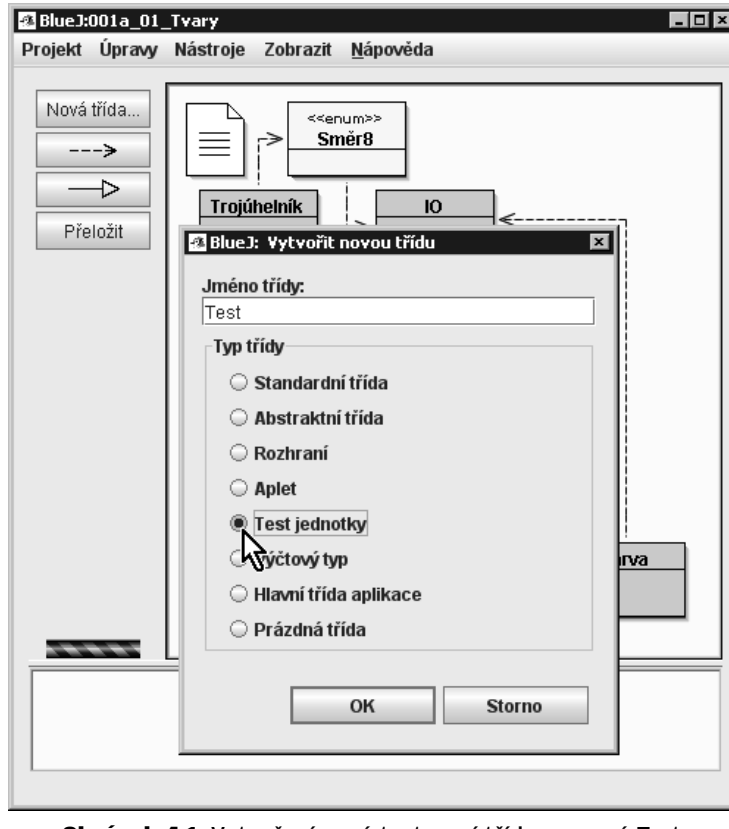
O vytvoření nové třídy požádáme např. stiskem tlačítka **Nová třída** na panelu tlačítek. *BlueJ* otevře dialogové okno, v němž musíš zadat, jak se bude nová třída jmenovat a jaký typ třídy má *BlueJ* vytvořit.

Jestli dovolíš, tak ještě chvíli odložím vysvětlování pravidel, podle nichž se tvoří názvy, a poradím ti, abys novou třídu nazvala **Test**. Přepínač **Typ třídy** pak nastav na **Test jednotky**, protože jsme si před chvílí řekli, že *BlueJ* je ochoten za nás naprogramovat jedině testovací třídu. Své zadání pak potvrď. *BlueJ* vytvoří novou třídu, pojmenuje ji `Test` a zobrazí ji ve volném prostoru v diagramu tříd.

Jak jsme si již řekli, testovací třída má mezi třídami zvláštní postavení. *BlueJ* na ně upozorňuje stereotypem `«unit test»` a stejně jako ostatním třídám označeným nějakým stereotypem ji přiřadil její vlastní barvu.

60. Stereotypem? Co je to?

Nesnaž se mi naznačit, že jsi větší sklerotik než já. O stereotypech jsme se bavili v pasáži *Třídy v projektu* na straně 46. Jako stereotyp označujeme krátký text uzavřený ve «francouzských uvozovkách», který je v diagramu tříd zapsán nad názvem třídy a naznačuje nějakou její zvláštnost – např. to, že se jedná o testovací třídu.



Obrázek 4.1: Vytvoření nové testovací třídy nazvané Test

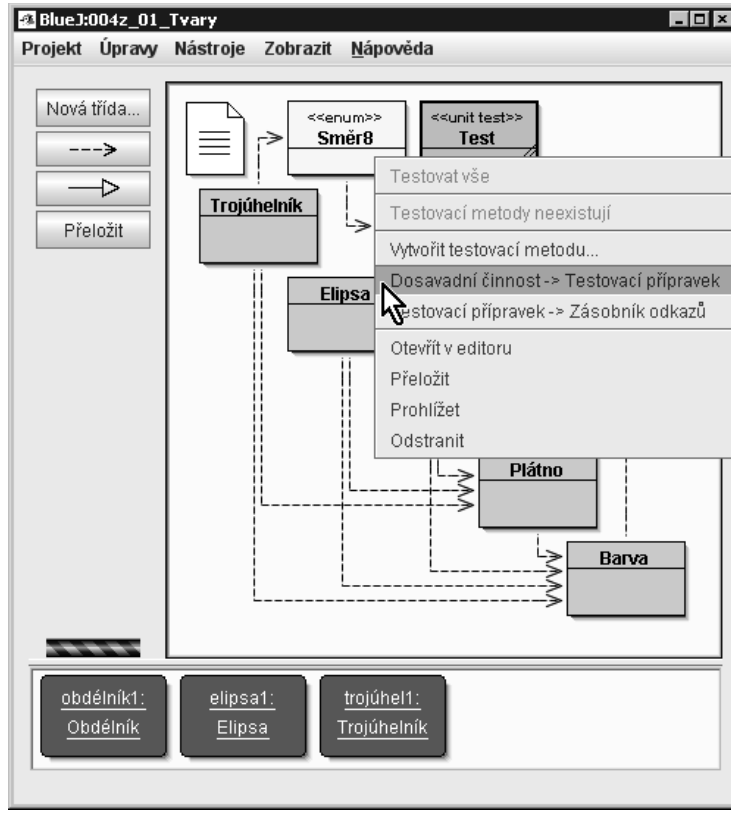
Vytvoření testovacího přípravku

61. Třídu mám. Takže hurá na přípravek! Co mám udělat?

Nic, co bys už neznala. Začneš tím, že restartuješ virtuální stroj a pak budeš posílat zprávy, které uvedou systém do stavu, ve kterém by se měly spouštět testy. *BlueJ* bude tvé počínání sledovat a na požádání je pak převede na program, který vytvoří přípravek a uvede program do požadovaného stavu.

62. Počkej, počkej. Nějak mu přece musím říct, že teď začínám ukazovat, jak vytvořit přípravek.

BlueJ si pamatuje všechny akce od posledního překladu nebo restartu virtuálního stroje. Začneš-li proto něco předvádět hned po překladu, nemusíš říkat nic. Budeš-li před předváděním činnosti, kterou si má *BlueJ* zapamatovat, provádět něco, co si *BlueJ* pamatovat nemá, musíš virtuální stroj nejprve restartovat.



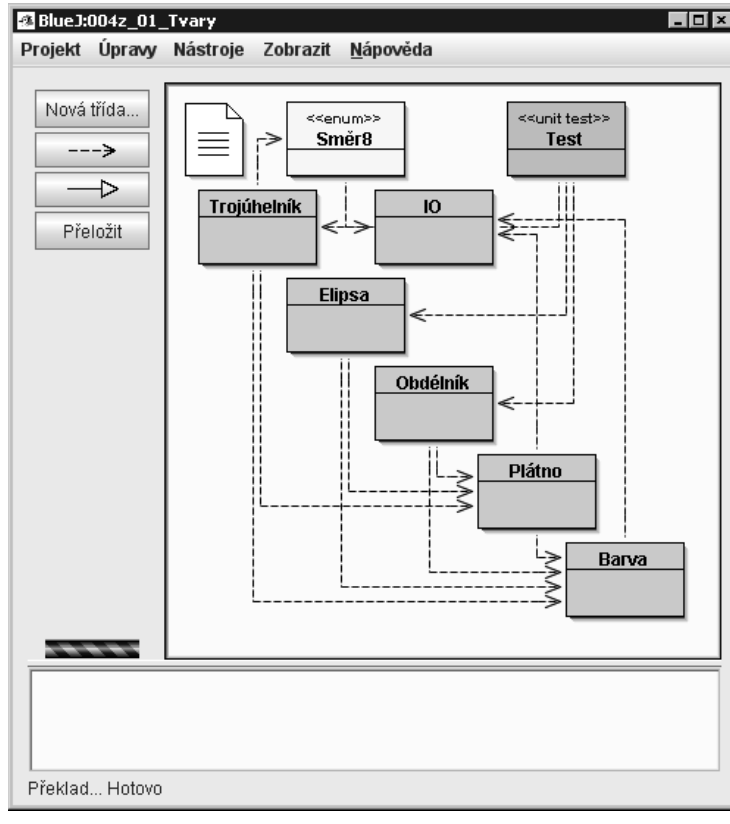
Obrázek 4.2: Žádost o převedení zapamatovaných akcí na program pro vytvoření testovacího přípravku

63. To chápu. A jak mu řeknu, že jsem skončila?

Pošleš *BlueJ* zprávu **Dosavadní činnost → Testovací přípravek**, kterou najdeš v místní nabídce testovací třídy. *BlueJ* pak vytvoří příslušný program. Jeho funkci můžeš vyzkoušet zasláním zprávy **Testovací přípravek → Zásobník odkazů**, po níž *BlueJ* vytvořený program spustí a naplní ti zásobník odkazů příslušnými odkazy – testovacím přípravkem.

64. Vytvořila jsem obdélník, elipsu a trojúhelník, s nimiž jsme si minule hráli. Když jsem požádala o vytvoření přípravku, tak se v diagramu tříd objevily nové šipky.

To je správně. *BlueJ* natáhl šipky závislosti od testovací třídy ke třídám grafických tvarů. Posuň testovací třídu kousek doprava a uvidíš, že od ní vedou šipky směřující ke třídám `Obdélník`, `Elipsa` a `Trojúhelník` (viz obrázek 4.3). Udělal to proto, že testovací třída poslala při tvorbě přípravku zprávy třídám grafických tvarů, v nichž je žádala o vytvoření nové instance. Tím se ale stala na těchto třídách závislá. Kdyby se totiž chování oslovených tříd jakkoliv změnilo, mohlo by to ovlivnit i chování naší testovací třídy (ostatně brzy si to ukážeme).



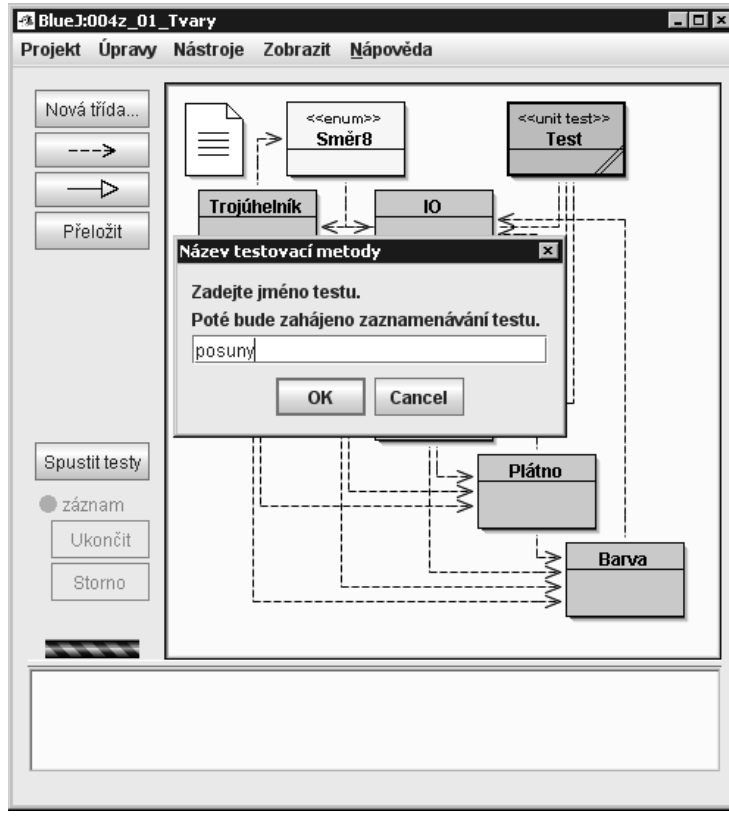
Obrázek 4.3: Závislosti testovací třídy po definici vytvoření přípravku

Vytvoření testu

65. Máš pravdu. Přípravek tedy mám. Jak nyní připravím nějaký test?

Doporučuji ti pro jistotu nejprve restartovat virtuální stroj, aby ses s *BlueJ* shodla na výchozím stavu programu před začátkem testu. Pak opět otevřeš místní nabídku testovací třídy (viz obrázek 4.2) a zadáš příkaz **Vytvořit testovací metodu.** Tím pošleš *BlueJ* zprávu, že mu chceš předvést postup, jak otestovat přípravek, který má tato třída na starosti.

BlueJ nechá testovací třídu vytvořit přípravek a na panelu tlačítek zobrazí tlačítko **Spustit testy**, kterým můžeš v budoucnu spouštět všechny testy, spolu s indikátorem **Záznam** (ten se „rozsvítí“ při záznamu testu) a tlačítky **Ukončit** a **Storno** pro ukončení, resp. stornování předváděného testu (i ta „ožijí“ až při spuštění testu). Současně otevře dialogové okno, v němž se tě zeptá, jak chceš vytvářený test pojmenovat (viz obr. 4.4).



Obrázek 4.4: Dotaz na název testu

Dohodněme se, že jej pojmenuješ **posuny.** Jakmile zadáš a potvrdíš název testu, začne *BlueJ* zaznamenávat, co děláš. Rozsvítí indikátor **Záznam**, který bude červeným „světýlkem“ upozorňovat na zaznamenávání příkazů testu. Současně „ožijí“ obě tlačítka pro ukončení testu.

Kdyby sis náhodou uprostřed předvádění uvědomila, žes udělala něco špatně, stiskneš tlačítko **Storno** a můžeš začít znovu.

Běží-li ale všechno správně, tak po ukončení testu stiskneš tlačítko **Ukončit**. *BlueJ* pak „zhasne" červené světýlko, převede zapamatované akce na program, který vloží do zdrojového kódu testovací třídy. Do místní nabídky testovací třídy pak zařadí povel, kterým tento program vyvoláš.

Vše, o čem jsme si tu povídali, jsem se opět pokusil předvést v doprovodné animaci. Pusť si ji a vše si s ní ještě jednou vyzkoušej.

Vytvoření testovací třídy a jejího přípravku: `OOPNZ_104_A1_TestovaciTrida`
Animace předvede, jak se vytváří nová testovací třída, jak se interaktivně definuje tvorba testovacího přípravku a jak je možno interaktivně zadat testy, které s tímto přípravkem pracují.

66. Všechno chodilo. Jenom mi bylo divné, proč při každém přesunu obrazce zůstala na jeho původní pozici nevybarvená oblast, když pod ním byl jiný objekt.

Objekty se přesouvají stejně, jako kdyby byly nakreslené na papíře. Nejprve se musí v původní pozici smazat a pak se nakreslí v nové pozici. Protože ale objekty o sobě nevědí, tak nemohou objektům, které leží pod nimi, poslat zprávu, aby se překreslily. To zatím musíš zabezpečit ty tím, že odmazané objekty o překreslení sama požádáš.

Schválně jsem to udělal takto jednoduše, a mohli bychom říci i nešikovně, abys měla motivaci, až budeme přecházet k chytřejšímu plátnu. Pro ten přechod se totiž budeš muset naučit několik dalších konstrukcí a zákonitostí.

67. Znám celou řadu jednoduše ovládaných programů, které se takto hloupě nechovají, a k jejich ovládání nic dalšího znát nemusím.

Samozřejmě, že bych mohl program upravit, aby se nechoval takto hloupě. Naším cílem ale není hýbat s objekty. My se tu učíme programovat. Kdybych program upravil, aby se již nyní choval chytře, musel bych před tebou skrýt některé konstrukce, které mají toto chytré chování na starosti, protože bys je nyní ještě nechápala. Až bychom je pak potřebovali použít, bylo by ti divné, k čemu jsou dobré, když jsi je do té doby nepotřebovala.

Vydrž to ještě chvilku, už to nebude dlouho trvat.

Metody a konstruktory

68. Zkusím to. Ještě bych měla jeden dotaz. Pořád říkáš, že posílám objektům zprávy, ale v místní nabídce bylo, že chceme po *BlueJ*, aby vytvořil testovací metodu. Co je to ta metoda.

Metoda je část programu, která má na starosti reakci objektu na danou zprávu. Každé zprávě, jíž objekt rozumí, je přiřazena metoda, která se postará o příslušnou reakci. Každé poslání zprávy se tak v programu převede na volání příslušné metody. Dokud není pro nějaký objekt (třídu, instanci) definována patřičná metoda, nemůže tento objekt přijímat odpovídající zprávy.

Až budeme psát programy, budeme definovat třídy a ve třídách jejich metody. Prozatím za nás metody definovalo prostředí *BlueJ*. My jsme mu něco předvedli a ono definovalo metodu, která předvedenou akci uměla na požádání zopakovat.

Programátoři pak většinou nehovoří o *posílání zpráv*, ale o *volání metod*. Oba termíny jsou vlastně synonyma obdobně jako *objekt* a *instance*. Když se pohybujeme na abstraktnější úrovni popisu problému, používáme spíše termín *zaslání zprávy*, pohybujeme-li se v kódu, dáváme přednost termínu *volání metody*.

I program, který vytváří přípravek, je metodou. Když tedy posíláme zprávu, aby třída spustila náš test, *BlueJ* ve spolupráci s testovací třídou zabezpečí, aby se nejprve zavolala metoda, která vytvoří přípravek, a po ní pak metoda, která zopakuje náš test.

Když už jsem se zmínil o metodách, měl bych ti ještě prozradit, že metody, které mají na starosti konstrukci nového objektu, se jmenují **konstruktory**. Konstrukce objektu je dvoufázová záležitost:

1. Nejprve je virtuálnímu stroji poslána zpráva `new Xxx`, kde `Xxx` je název třídy, jejíž instanci chceme vytvořit. Virtuální stroj vyhradí místo v paměti a připraví některé systémové komponenty, které jsou potřeba pro správnou funkci každého objektu.
2. Poté je zavolána speciální metoda nazývaná **konstruktor**, která ve vyhrazené paměti vytvoří (zkonstruuje) objekt dané třídy a vrátí volajícímu programu odkaz na právě vytvořený objekt.

69. Je mi to divné. Řekla bych, že hlavní zásluhu na konstrukci objektu má virtuální stroj, který najde a připraví paměť, v níž bude zkonstruovaný objekt bydlet.

Konstrukci objektu většinou přirovnávám k výrobě hrnečku či džbánku. Paměť, kterou zajišťuje virtuální stroj, je něco jako hlína. Dokud nemáme hlínu, nemůžeme žádný hrnek vyrobit. Vlastní tvorbu hrnku má však na starosti hrnčíř, který jej vytvaruje podle svých představ. Tohoto hrnčíře zastupuje v programu konstruktor. Ten upraví přidělenou paměť do takové podoby, aby pak mohla sloužit jako požadovaný objekt.

Opakování

Zopakujme si, co jsme se v této lekci dozvěděli:

- Prostředí *BlueJ* je schopno sledovat naše počínání, převést je na program a ten uložit do testovací třídy.
- Protože má testovací třída z hlediska prostředí *BlueJ* nestandardní vlastnosti (*BlueJ* je schopen ji naprogramovat sám), je v diagramu tříd její název doplněn stereotypem `«unit test»` a její obdélník je vybarven jinou barvou než běžné třídy.
- Stereotyp je krátký text uzavřený ve «francouzských uvozovkách», který je v diagramu tříd zapsán nad názvem třídy a naznačuje nějakou její zvláštnost – např. to, že se jedná o testovací třídu.
- Testovací třídy většinou obsahují sadu několika testů pracujících nad stejnou výchozí sadou objektů. Tuto výchozí sadu objektů označujeme jako *testovací přípravek*. Každá testovací třída má vlastní přípravek.

- Testovací přípravek můžeme vytvořit tak, že po překladu tříd či restartu virtuálního stroje předvedeme, jak by se měl vytvářet. Po příkazu **Dosavadní činnost → Testovací přípravek** v místní nabídce testovací třídy, pro jejíž testy přípravek vytváříme, *BlueJ* definuje na základě předvedené činnosti metodu, kterou bude spouštět před každým testem z této třídy.
- Program vytvářející testovací přípravek můžeme spustit i samostatně zadáním příkazu **Testovací přípravek → Zásobník odkazů**. *BlueJ* pak naplní zásobník odkazů odkazy na objekty z přípravku.
- O vytvoření testu požádáme zadáním příkazu **Vytvořit testovací metodu** v místní nabídce testovací třídy. *BlueJ* se nejprve zeptá na název vytvářeného testu a pak zobrazí na panelu tlačítek tlačítka pro ukončení a stornování testu spolu s indikátorem záznamu prováděných akcí.
- Předvádění ukončíme stiskem tlačítka **Ukončit** na panelu tlačítek. *BlueJ* pak definuje testovací metodu, která dokáže zopakovat předvedenou činnost, a do místní nabídky příslušné testovací třídy přidá příkaz, který tento test vyvolá.
- Metoda je část kódu definující reakci objektu na danou zprávu. Termíny *zaslání zprávy* a *volání metody* jsou vzájemně ekvivalentní.
- Tvorba objektu probíhá ve dvou fázích:
  - nejprve virtuální stroj vyhradí potřebné místo v paměti,
  - pak se v tomto místě vytvoří požadovaný objekt a žadateli se vrátí odkaz na něj.
- Metoda, která má na starosti konstrukci objektu ve vyhrazené paměti a předání odkazu na zkonstruovaný objekt, se nazývá **konstruktor**.

Projekt:
Výsledná podoba projektu, k níž jsme se dopracovali na konci lekce, je v projektu `104z_01_Tvary`.