DUM č. 7 v sadě

35. Inf-11 Objektové programování v Greenfoot

Autor: Lukáš Rýdlo

Datum: 27.06.2014

Ročník: studenti semináře

Anotace DUMu: Závěrečná lekce prvního projektu - dokumentace, generování nových objektů ve scéně pomocí třídy Generator a zavedení počítadla populace.

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
2007-13 OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Projekt: Simulace života zajíce v Greenfoot
Počítadla a generátory

Úvod

Závěrečná lekce projektu simulátoru života zajíců a lišek je věnována především zhodnocení a shrnutí projektu. Funkční rozšíření bude spočívat pouze ve zvýšení uživatelské přívětivosti aplikace tím, že zavedeme možnost přímo za běhu aplikace počítat stavy populace a vkládat nebo odebírat lišky a zajíce.

Studenti by měli projekt zhodnotit, specifikovat, co jim dělalo obtíže a co jim naopak přišlo jednoduché, jaká pozitiva nebo negativa užití prostředí Greenfoot přineslo. Navazující projekt implementace téhož ve strukturovaném jazyce C s knihovnou Allegro totiž vyzdvihne výhody předpřipraveného grafického prostředí, ale snad také poukáže na bližší spojení programátora s kódem a snazší pochopení struktury grafického programu.

Nezbytně nutné je studenty přimět kód okomentovat komentáři v JavaDocu. V průběhu programování jsme kód nekomentovali, což je podstatná chyba. Je možné zadat doplnění dokumentace s určitým zpožděním a tím studentům názorně ukázat, že kód, který oni sami před třemi týdny vymysleli a napsali, jim samotným může dělat nyní problémy pochopit. Zdůrazníme význam automaticky generované dokumentace (JavaDoc nebo Doxygen) ze speciálně formátovaných komentářů.

Úkoly s řešeními

1. Vytvořte novou třídu Generator. Jako ikonu/obrázek bude mít v kódu vykreslený zelený kruh s poloměrem 30 px a přes něj vždy obrázek zajíce, zaječky, lišky nebo mrkve podle aktuální volby, co bude vkládat. Poslední možností bude zobrazení červeného kruhu, který bude znamenat, odstranění objektu ze scény. Volba se provede kliknutím na tento objekt, čímž se bude cyklicky střídat mezi zmíněnými volbami. Funkci voleb zatím neimplementujte, pouze vytvořte toto „tlačítko“ reagující na kliknutí.

   Na načítané obrázky je vhodné vytvořit pole objektů GreenfootImage a přiřadit je hned v konstruktoru. Ikonu může kreslit samostatná funkce, volaná vždy při změně. K samotné změně dochází událostí kliknutí myši, kterou dokáže odchytit uvnitř metody act vhodná statická metoda třídy Greenfoot. Viz vzor s vynechávkami:

```
import greenfoot.*;
import □.□.Color;

public class Generator □ □
{
    private GreenfootImage icon = □ □(60,60);
    private □ option = 0;
    private GreenfootImage□ images = new GreenfootImage[□];

    public Generator() {
        this.images[0] = new GreenfootImage("zajicR1.png");
        this.images[1] = new GreenfootImage("zajicR2.png");
        this.images[2] = new GreenfootImage("Carrot.png");
        this.images[3] = new GreenfootImage("liskaR1.png");
        for (int i = 0; i<4; i++) this.images[□].□(50, 50);
        this.redrawIcon();
        this.setImage(□);
    }

    public void redrawIcon() {
        this.icon.□();
        if (option <4) {
            this.icon.setColor(Color.□);
            this.icon.fillOval(0,0,60,60);
            this.icon.drawImage(this.□[this.□], 5, 5);
        }
        if (option ==4) {
            this.icon.setColor(Color.□);
            this.icon.fillOval(0,0,60,60);
        }
    }

    public void act()
    {
```

```
            if(Greenfoot.□(□)) {
                this.option++;
                this.option □=5;
                this.redrawIcon();
            }
        }
}
```

2. V třídě Generator implementujte v metodě act vkládání instance příslušné třídy podle aktuálně nastaveného volby (obrázku) po kliknutí na herní plán (volné místo v herním plánu) do místa kliknutí. Pokud je výběr nastavený na mazání (červené kolečko), pak detekujte kliknutí na některého hráče v plánu a po kliknutí jej odstraňte ze světa. Tato funkcionalita bude součástí třídy Generator nikoliv jednotlivých hráčů a nesmí odstranit ze scény třídu Generator.

   Implementace je překvapivě poměrně jednoduchá a sestává z podmínky pro vkládání a podmínky pro mazání. V prvním případě se použije vhodná metoda třídy Greenfoot pro získání kliknutí do herního plánu, v druhém případě je situace o něco komplikovanější, protože bude nutné získat navíc i odkaz na instanci třídy, na kterou bylo kliknuto. Také nesmíme zapomenout, že některé získané reference mohou mít hodnotu null, což znamená, že na nich pak nelze volat metody.

```
if (Greenfoot.□(this.□())) {
   int x = Greenfoot.□().□();
   int y = Greenfoot.□().□();
   try{
      switch(this.option) {
      case 0: //rabbit male
         this.□().□(new Rabbit(Rabbit.□), x, y);
         break;
      case 1: //rabbit female
         this.□().□(new Rabbit(Rabbit.□), x, y);
         break;
      case 2: //carrot
         this.□().□(new Carrot(□), x, y);
         break;
      case 3: //fox
         this.□().□(new Fox(), x, y);
         break;
      }
   } catch (□ e) {
   }
}

MouseInfo mouse = Greenfoot.□();
if (this.option==4 □ mouse != □ □
    mouse.□()==1 && mouse.□() != null &&
    !(mouse.□() □ Generator)) {
   this.getWorld().□(mouse.getActor());
}
```

3. Vytvořte třídu Counter, která bude počítat populaci zajíců a lišek a bude se vkládat do scény jako hráč. Místo obrázku bude bílý – částečně průhledný – obdélník o vhodných rozměrech a bude vypisovat v textové podobě přes bílý obdélník informace o celkovém počtu zajíců, počtu dospělých samců a samic a souhrnném počtu nedospělých jedinců. Dále vypíše počet lišek.

   Řešení obsahuje jedinou záludnost spočívající v počítání jednotlivých pohlaví zajíců. Jelikož nejsou rozlišeny třídou, ale atributem, bude nutné „ručně“ projít seznam všech zajíců a přepočítat počty sekvenčním průchodem seznamu. Řešení je kompletní, až na vynechané metody, názvy tříd, balíků a proměnných:

```
import greenfoot.*;
import □.□.List;
import □.□.Color;

public class Counter extends Actor
{
    private GreenfootImage i = new □(200,40);

    public Counter() {
        this.□(this.□);
    }

    public void act()
    {
        List<□> rabbits = this.getWorld().□(□);

        int male = 0;
        int female = 0;
        int child = 0;

        for(□ rabbit : rabbits) {
            switch(rabbit.□()) {
                case □.MALE:
                   male++;
                   break;
                case □.FEMALE:
                   female++;
                   break;
                case □.CHILD_MALE:
                case □.CHILD_FEMALE:
                   child++;
                   break;
            }
        }

        this.i.□();
        this.i.setColor(new □(255,255,255,128));
        this.i.fill();
        this.i.setColor(Color.black);
```

```
        this.i.□("Rabbits: "+rabbits.size()+
                 " (M: "+male+", F: "+female+", CH: "+child+")",
                 5, 15);
        this.i.□("Foxes: "+this.getWorld().□(□).size(), 5, 34);
    }
}
```

4. BONUS: Pokročilejší studenti nechť zkusí vytvořit třídu Graph, která bude v konstruktoru přebírat třídu (class), maximální předpokládaný počet a šířku a výšku obrázku. Třída vytvoří podle zadaných rozměrů obdélník vyplněný částečně průhlednou bílou a v každém actu bude počítat počet tříd předaného typu. Pak vykreslí graf. Graf bude sloupcový (za každý jeden záznam se vytvoří barevný sloupeček o šíři 1 px) a vykreslovaný od konce, tzn. poslední hodnota bude vždy na konci – posledním sloupci – grafu, starší hodnoty se budou překreslovat před ni. Počet pamatovaných hodnot je dán šířkou grafu. Přesáhne-li hodnota maximální předpokládaný počet, vykreslí se sloupeček v maximální velikosti a červenou barvou.

   Úkol není zcela jednoduchý, naráží na problém, jak si pamatovat starší hodnoty. Jednou možností je používat k tomu pouze obrázek a grafické operace, což není vhodné. Lepší bude najít nějakou vhodnou datovou strukturu, což vzhledem k charakteru grafu je na první pohled fronta. Vybereme si vhodnou implementaci a použijeme nejspíš kolekci LinkedList, která umožňuje vkládat na konec, odebírat zepředu metodou i procházet postupně prvky v celé frontě. Jelikož se jedná o bonus, vzorové řešení neposkytuji.

5. Doplňte detailní dokumentaci ke všem třídám, metodám a atributům v projektu. Nezapomeňte popisovat všechny přebírané parametry a návratové hodnoty. Použijte JavaDoc a snažte se psát anglicky. Lepší špatná angličtina než dobrá čeština, které nikdo v zahraničí nerozumí.

   Tento úkol je poměrně dost rozsáhlý, protože jsme vytvořili s osmi třídami a řadou metod a parametrů. Je vhodné jej zadat jako domácí úkol a pak pouze zkontrolovat výslednou dokumentaci (použitou JavaDoc syntaxi). Krátký český úvod do JavaDoc je na stránce http://cs.wikibooks.org/wiki/Java/Javadoc, další vhodné a detailnější materiály v angličtině jsou například na stránkách:

   http://students.cs.byu.edu/~cs240ta/fall2012/tutorials/javadoctutorial.html
   http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

6. Vyexportujte celý projekt jako jar soubor.

Úkoly

1. Vytvořte novou třídu Generator. Jako ikonu/obrázek bude mít v kódu vykreslený zelený kruh s poloměrem 30 px a přes něj vždy obrázek zajíce, zaječky, lišky nebo mrkve podle aktuální volby, co bude vkládat. Poslední možností bude zobrazení červeného kruhu, který bude znamenat, odstranění objektu ze scény. Volba se provede kliknutím na tento objekt, čímž se bude cyklicky střídat mezi zmíněnými volbami. Funkci voleb zatím neimplementujte, pouze vytvořte toto „tlačítko“ reagující na kliknutí.
2. V třídě Generator implementujte v metodě act vkládání instance příslušné třídy podle aktuálně nastaveného volby (obrázku) po kliknutí na herní plán (volné místo v herním plánu) do místa kliknutí. Pokud je výběr nastavený na mazání (červené kolečko), pak detekujte kliknutí na některého hráče v plánu a po kliknutí jej odstraňte ze světa. Tato funkcionalita bude součástí třídy Generator nikoliv jednotlivých hráčů a nesmí odstranit ze scény třídu Generator.
3. Vytvořte třídu Counter, která bude počítat populaci zajíců a lišek a bude se vkládat do scény jako hráč. Místo obrázku bude bílý – částečně průhledný – obdélník o vhodných rozměrech a bude vypisovat v textové podobě přes bílý obdélník informace o celkovém počtu zajíců, počtu dospělých samců a samic a souhrnném počtu nedospělých jedinců. Dále vypíše počet lišek.
4. BONUS: Pokročilejší studenti nechť zkusí vytvořit třídu Graph, která bude v konstruktoru přebírat třídu (class), maximální předpokládaný počet a šířku a výšku obrázku. Třída vytvoří podle zadaných rozměrů obdélník vyplněný částečně průhlednou bílou a v každém actu bude počítat počet tříd předaného typu. Pak vykreslí graf. Graf bude sloupcový (za každý jeden záznam se vytvoří barevný sloupeček o šíři 1 px) a vykreslovaný od konce, tzn. poslední hodnota bude vždy na konci – posledním sloupci – grafu, starší hodnoty se budou překreslovat před ni. Počet pamatovaných hodnot je dán šířkou grafu. Přesáhne-li hodnota maximální předpokládaný počet, vykreslí se sloupeček v maximální velikosti a červenou barvou.
5. Doplňte detailní dokumentaci ke všem třídám, metodám a atributům v projektu. Nezapomeňte popisovat všechny přebírané parametry a návratové hodnoty. Použijte JavaDoc a snažte se psát anglicky. Lepší špatná angličtina než dobrá čeština, které nikdo v zahraničí nerozumí.
6. Vyexportujte celý projekt jako jar soubor.