DUM č. 3 v sadě

35. Inf-11 Objektové programování v Greenfoot

Autor: Lukáš Rýdlo

Datum: 08.06.2014

Ročník: studenti semináře

Anotace DUMu: Třída Mrkev. Implementace vegetačního cyklu mrkve. Změna (obarvení) obrázku podle fáze růstu. Znázornění kusů a času, vlastní progress bar.

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
Třída mrkev

Úvod

V této lekci vytvoříme v prostředí Greenfoot třídu Carrot (mrkev) a implementujeme její fáze růstu a možnost sežrání. Pro znázornění fází zrání, zralosti a hnití necháme obrázek mrkve měnit barvu od zelené po oranžovo-červenou až do hnědavé.

Tato lekce nevyžaduje žádné zvláštní znalosti, ale přináší spoustu „řemeslných“ úloh a je příležitostí procvičit (a zkontrolovat) vhodný programátorský styl zápisu kódu, precizní a systematické řešení problému.

Splnění všech úkolů (a tím vytvoření kompletní funkční třídy) je časově poměrně náročné, ale nemělo by průměrně zdatným studentům způsobit významnější problémy. Bude potřeba pravděpodobně studenty popostrčit při řešení matematických problémů se spočtením některých hodnot, ale je žádoucí, aby na řešení přišli žáci sami. V principu jde o aplikaci trojčlenky.

Poměrně důležitá je fáze analýzy třídy a rozvržení metod, které bude potřeba implementovat. Můžeme začít slovním zadáním:

Třída mrkve prochází 3 fázemi růstu: zrání, kdy mění postupně barvu od zelené po oranžovou a nelze mrkev jíst; zralosti, kdy je mrkev poživatelná a lze ji tedy jíst; hniloby, kdy mění barvu z oranžové na hnědou a jíst ji již nelze. Objekt reprezentuje více mrkví, jejichž počet je daný hodnotou předanou v konstruktoru. Jí se po jedné mrkvi ve fázi zralosti a zbývající počet mrkví i čas, po který lze jíst, budou znázorňovat dva ubíhající proužky v horní části obrázku. Maximální počet mrkví se nastavuje pouze v konstruktoru, celkový počet lze měnit voláním setteru, který povolí nastavení v rozsahu od 0 po definovaný maximální počet. Pro získání aktuálního počtu bude k dispozici getter a metoda udávající procentuální množství zbývající mrkve. Třída bude mít také metody vracející booleovské hodnoty pro dotaz na jednotlivé fáze růstu a metodu vracející v procentech v jaké části aktuální fáze růstu se objekt nachází. V rámci posílení programátorských návyků je vhodné striktně (vzorově) vyplňovat dokumentaci třídy a jejích metod – JavaDoc. Před započetím práce je nakreslíme UML diagram třídy, lze k tomu použít kromě tabule například volně šiřitelný program Umbrello (http://umbrello.kde.org), který umí i generovat kód nebo naopak generovat diagramy z kódu. Studenti pracují podle diagramu (jako kodéři, kteří dostali zadání od analytika).

Carrot
- phase : int - count : int - maxCount : int - liveTime : int - imageBuffer : GreenfootImage - imageCarrot : GreenfootImage
+ Carrot(maxCount : int) + getCarrotColor() : Color + getPhaseTimePercent() : int + addToPhase() + isGrowing() : boolean + isEatable() : boolean + isRotting() : boolean + redrawImage() + setCount(count : int) : boolean + getCount() : int + getPercentageCount() : int + getMaxCount() : int + eatOne() : boolean + act()

Alternativou je třeba yED (http://www.yworks.com/en/products_yed_about.html). Při nejasnostech s UML doporučuji příslušnou kapitolu své diplomové práce: Lukáš Rýdlo, Programovací jazyky pro výuku programování na SŠ, Masarykova Univerzita, Brno 2012 (dostupné z internetu přes www.theses.cz nebo přímo http://is.muni.cz/th/139514/fi_m/dp.pdf).

Úkoly s řešeními

1. Vytvořte třídu Carrot s novým obrázkem mrkve o rozměrech přibližně 50 × 50 px. Zjistěte, jaké jsou možnosti přebarvování vnitřku mrkve na různé barevné odstíny. Umí třída GreenfootImage nebo jiné třídy, které z ní lze dostat (třeba metodou getAwtImage()) přebarvit oblast s danou barvou? Pokud ne, jak by bylo možné tento problém obejít (zvažte, že některé grafické formáty – PNG nebo GIF – umí průhlednost)?

   

   Neexistuje žádná metoda třídy GreenfootImage ani BufferedImage ani Graphics, která by dokázala vyplňovat souvislou oblast barvou. Navíc je tento algoritmus poměrně časově náročný (jak uvidíme v projektu BMP editoru). Ale zmíněná průhlednost by se využít dala, pokud by oblast, která má být přebarvená, byla průhledná. Pak bychom vytvořili obrázek celý vyplněný požadovanou barvou a přes něj překryli obrázek s mrkví, který bude v místě obarvení průhledný (viz „bílá“ oblast na obrázku) a tudíž bude mít původní barvu pozadí… Takto upravovaný obrázek se ale bude muset měnit v kódu třídy.

2. Do třídy Carrot doplňte atributy podle UML diagramu. Mínus znamená, že jsou všechny atributy „private“ (+ je public, # je protected). Atribut phase určuje číselně fázi růstu, inicializujte na 0. Atributy count a maxCount jsou aktuální a maximální počet mrkve. Nemusíte inicializovat, musí se nastavit v konstruktoru. Atribut liveTime určuje délku „životního cyklu“ mrkve, tedy počet vykonání metody act během jedné trojice zrání-zralost-hnití. Tento atribut je final (neměnná konstanta) a nastavte ji na 800. Atribut imageBuffer slouží pro kreslení aktuálního obrázku. Přiřaďte mu novou instanci GreenfootImage o velikosti 50 × 50 px. Atribut imageBuffer v sobě nese načtený obrázek mrkve (přiřaďte přímo v deklaraci).

   Všechny atributy jsou celočíselné, tedy int. Není důvod používat objektový Integer. Třída GreenfootImage má několik konstruktorů, jeden z nich přebírá 2 integery určující rozměry nově vzniklého prázdného obrázku (kupř. `new GreenfootImage(10,10);` vytvoří čistý obrázek 10 × 10 px), jiný přebírá název resp. adresu obrázku (`new GreenfootImage("o.png");` vytvoří obrázek podle souboru „o.png“ v podadresáři images aktuálního projektu).

3. Vytvořte konstruktor, který přebírá jeden argument – počet mrkví na políčku (viz UML diagram). Pokud je počet nekorektní, nastavte jej na 1 a třídu vytvořte. Je potřeba nastavit v konstruktoru i jiné atributy? Žádný jiný (bezparametrický) konstruktor nevytvářejte.

   Konstruktor je výjimka, která nemá žádný návratový typ, ale měl by být public. Je potřeba nastavit kromě maxCount i count, ale není to nutné, protože se stejně musí nastavovat během fáze růstu. Konstruktor vypadá třeba takto (doporučuji upozornit na možnost použití ternárního poerátoru ?: místo podmínky if):

```
public Carrot(int maxCount) {
    this.maxCount = (maxCount>0?maxCount:1);
    this.count = this.maxCount;
}
```

4. Vytvořte settery a gettery setCount, getCount, getMaxCount, getPercentageCount podle UML diagramu. Nezapomeňte v setteru setCount kontrolovat správnost zadané hodnoty a vracet booleovské návratové hodnoty podle toho, zda byla nebo byla hodnota přiřazena.

   V následujícím vzorovém řešení jsou vynechané dva názvy a jedno číslo ve výpočtu procent (patří na místo symbolu □).

```
public boolean setCount(int count) {
    if (count>0 && count<this.maxCount) {
        this.count = count;
        return true;
    }
    return false;
}

public int getCount() {
    return this.count;
}

public int getPercentageCount() {
    return (this.□*□)/this.□;
}

public int getMaxCount() {
    return this.maxCount;
}
```

5. Vytvořte metodu addToPhase, která bude připočítávat jedničku k proměnné phase, ale tak, aby v proměnné byla vždy hodnota v rozmezí 0 až liveTime−1 (tedy 799). Místo podmínky if k tomu použijte vhodnou matematickou operaci (nápověda: když dělím číslem 4, dostávám jako zbytky po dělení 0 až 3). Vytvořte testy isGrowing (true, je-li fáze růstu), isEatable (true, je-li fáze zralosti) a isRotting (true, je-li fáze hniloby). Dodržte, že první čtvrtinu času rostlina roste (ostrá nerovnost), druhou a třetí čtvrtinu je poživatelná (neostrá nerovnost u obou hranic) a poslední čtrtinu hnije (neostrá nerovnost). Používejte striktně konstantu liveTime a její násobení či dělení, místo konkrétních čísel (aby bylo možné dodatečně změnit činnost pouhou úpravou konstanty) – v těchto metodách stačí jen return a vhodně zapsaná nerovnost. Až budete mít metodu isGrowing, využijte ji v addToPhase k doplnění kódu, který během „addToPhase“ nastaví počet mrkví na maximální možný, pokud je mrkev ve fázi růstu. Vytvořte metodu getPhaseTimePercent, která bude vracet počet procent uběhnutého času v dané fázi růstu. Uvědomte si, že je nutné počítat zvlášť (if) pro každou ze tří fází. Nalezněte přepočet, jak zařídit, že např. číslo 200 je 0 % a 600 je 100 % fáze zralosti (viz pravidla pro fáze výše).

   Řešení je kratší než zadání, ale je potřeba zkontrolovat, že se nepoužívají zbytečně složité konstrukce podmínek v místech, kde to není třeba, a že jsou správně vymyšlené výpočty. Vzorové řešení opět obsahuje záměrně vynechané části výpočtů, symbol □ nahrazuje proměnné a čísla. Znak ▒ nahrazuje operátor zbytku po celočíselném dělení.

```
public int getPhaseTimePercent() {
    if (this.isGrowing()) return (□*□)/(□/4);
```

```
    if (this.isEatable()) return (□-(□/4))*□/(□/2);
    return (□-(□*3/4))*100/(□/□);
}

public void addToPhase() {
    this.phase++;
    phase ▒= this.liveTime;

    if (this.isGrowing()) {
         this.count = this.maxCount;
    }
}

public boolean isGrowing() {
    return □ < □/4;
}

public boolean isEatable() {
    return phase >= □/4 && phase <=□*3/4;
}

public boolean isRotting() {
    return □ > □*□/□;
}
```

6. Vytvořte metodu redrawImage, která bude vykreslovat nový obrázek mrkve. Používejte atribut imageBuffer, který vyplníte nejprve oranžovou barvou a potom přes něj překreslíte obrázek z imageCarrot. Nezapomeňte na závěr vhodnou metodou třídy Actor (kterou Carrot podědila) nastavit obrázek imageBuffer jako obrázek třídy. Barvy se v Greenfootu volí pomocí třídy java.awt.Color, kterou musíte naimportovat. Třída má buď konstruktor, kde lze nastavit červenou, zelenou a modrou složku nebo statické atributy, které obsahují některé přednastavené barvy. Vizte dokumentaci: http://docs.oracle.com/javase/7/docs/api/java/awt/Color.html

   Řešení je jednoduché: `this.imageBuffer.fill()` *obrázek vyplní, předtím se musí ale jinou metodou místo „fill“ nastavit barva a poté se další metodou, která kreslí obrázek, vykreslí do imageBufferu imageCarrot – požadovaná dvě čísla jsou souřadnice pro umístění obrázku a jsou tedy 0, 0. Na závěr se this.imageBuffer předá jednomu ze setterů třídy Actor, který nastavuje obrázek:* `this.set□(this.imageBuffer);`

7. Doplňte do redrawImage kreslení obdélníčku v horní části, který se kreslí pouze ve fázi zralosti a zobrazuje zeleným pruhem procentuální množství zbylých mrkví a pod ním žlutý (nebo oranžový) procentuální zbývající čas, po který bude možno jíst (viz obrázek). Černý obdélníček je široký 42 a vysoký 6 px a je umístěný 4 px zleva a 1 px shora. Zelený pruh je zleva 5 px a shora 2 px, široký nejvýše 40 a vysoký 2 pixely. Podobně žlutý, který je jen níž (shora 4 px).

   Ukázkový kód opět obsahuje vynechané části názvů metod, které patří třídě GreenfootImage, takže je lze dohledat v její dokumentaci, resp. po stisku ctrl+enter za tečkou za imageBufferem je prostředí Greenfoot samo nabídne. Podobně jsou vynechaná některá čísla, která jistě musí každý

student zvládnout spočítat sám (procentuální poměr délky rámečku). Tento snippet (úsek kódu) pochopitelně patří umístit za vykreslení obrázku s mrkví, ale před nastavení bufferu jako obrázku třídy. Studenti budou mít možná tendenci zkoušet, jestli metoda funguje a budou se divit, že s nic nepřekresluje. To je dáno tím, že tuto metodu zatím nikde nevolají – musí její volání umístit do metody act (viz následující úkol).

```
if(this.isEatable()) {
    this.imageBuffer.setColor(Color.black);
    this.imageBuffer.□Rect(4, 1, 42, 6);
    this.imageBuffer.setColor(Color.green);
    this.imageBuffer.□Rect(5, 2, □*□/□,2);
    this.imageBuffer.setColor(Color.orange);
    this.imageBuffer.□Rect(5, 4, □*(□-□)/□,2);
}
```

8. Do metody act() doplňte volání dvou dříve naprogramovaných metod, které způsobí nejprve přepočítání fáze a pak překreslení obrázku. Doplňte také metodu eatOne(), která po zavolání sníží hodnotu count o 1, pouze pokud nějaká mrkev ještě zbývala a pokud je instance zrovna ve fázi zralosti (isEatable). Vracejte true nebo false, podle úspěšnosti (snědeno, nesnědeno).

   Triviální metoda act obsahuje pouze `this.addToPhase();` a `this.redrawImage();`, což ale způsobí, že budou vidět výsledky vykreslování obrázku. V metodě eatOne je jen podmínka (ve výrazu je isEatable a zároveň getCount()>0) a pak `this.count--; return true;` a za podmínkou `return false;`. Nepoužíváme zbytečně „else“ větev, return false je prostě vždy, pokud se k němu za ifem dostaneme.

9. Chybí vytvořit metodu getCarrotColor a v metodě redrawImage nastavovat barvu pro výplň právě na hodnotu, kterou bude vracet. Metoda vrací přímo instanci třídy Color (z balíku java.awt) a to tak, že použije vhodný konstruktor třídy Color, kterému předá červenou, modrou a zelenou složku barev. Ve fázi zralosti předávejte barvu s RGB hodnotami: 249, 117, 0 (oranžová). Ve fázi zrání předávejte barvu, která přechází podle procenta času (getPhaseTimePercent) od zelené (RGB hodnoty 32, 106, 35) po výše uvedenou oranžovou. Ve fázi hnití nechť barva postupně přechází od oranžové po hnědou (RGB hodnoty 93, 74, 34). Jak spočítáme odstíny barev mezi krajními barvami podle procenta času? K uložení čísel barev v metodě nepoužívejte samotné proměnné int, ale použijte vždy 3prvkové pole int pro každou barvu.

   Tato metoda je nejtěžší i po stránce úvahy nad číselnou hodnotou barvy na barevném přechodu. Pro představu můžeme situaci přirovnat k posunu po vektoru. Chceme-li plynule přecházet po vektoru, pak měníme každou složku (x a y) postupně zvlášť v poměru vzdálenosti mezi startem a cílem. Stejně tak i barva na lineárním gradientu přechází tak, že se v lineárním poměru postupně mění každá složka zvlášť (R, G a B) od počáteční po cílovou barvu. Fáze zrání a hniloby ošetříme nejprve pomocí podmínky, pak na konci metody už bez podmínky vždy vracíme oranžovou. Řešení opět vynechává rámečkem čísla, proměnné nebo metody.

```
private Color getCarrotColor() {
   int green[]  = {32,106,35};
   int orange[] = {249,117,0};
   int brown[]  = {94,73,34};
```

```
   if (this.isGrowing()) {
      return new Color(
         □[0]+(□[0]-□[0])*□/100,
         □[1]+(□[1]-□[1])*□/100,
         □[2]+(□[2]-□[2])*□/100
      );
   }
   if (this.isRotting()) {
      return new Color(
         □[0]+(□[0]-□[0])*□/100,
         □[1]+(□[1]-□[1])*□/100,
         □[2]+(□[2]-□[2])*□/100
      );
   }
   return  new Color(orange[0],orange[1],orange[2]);
}
```

Úkoly

1. Vytvořte třídu Carrot s novým obrázkem mrkve o rozměrech přibližně 50 × 50 px. Zjistěte, jaké jsou možnosti přebarvování vnitřku mrkve na různé barevné odstíny. Umí třída GreenfootImage nebo jiné třídy, které z ní lze dostat (třeba metodou getAwtImage()) přebarvit oblast s danou barvou? Pokud ne, jak by bylo možné tento problém obejít (zvažte, že některé grafické formáty – PNG nebo GIF – umí průhlednost)?

2. Do třídy Carrot doplňte atributy podle UML diagramu. Mínus znamená, že jsou všechny atributy „private“ (+ je public, # je protected). Atribut phase určuje číselně fázi růstu, inicializujte na 0. Atributy count a maxCount jsou aktuální a maximální počet mrkve. Nemusíte inicializovat, musí se nastavit v konstruktoru. Atribut liveTime určuje délku „životního cyklu“ mrkve, tedy počet vykonání metody act během jedné trojice zrání-zralost-hnití. Tento atribut je final (neměnná konstanta) a nastavte ji na 800. Atribut imageBuffer slouží pro kreslení aktuálního obrázku. Přiřaďte mu novou instanci GreenfootImage o velikosti 50 × 50 px. Atribut imageBuffer v sobě nese načtený obrázek mrkve (přiřaďte přímo v deklaraci).

3. Vytvořte konstruktor, který přebírá jeden argument – počet mrkví na políčku (viz UML diagram). Pokud je počet nekorektní, nastavte jej na 1 a třídu vytvořte. Je potřeba nastavit v konstruktoru i jiné atributy? Žádný jiný (bezparametrický) konstruktor nevytvářejte.

4. Vytvořte settery a gettery setCount, getCount, getMaxCount, getPercentageCount podle UML diagramu. Nezapomeňte v setteru setCount kontrolovat správnost zadané hodnoty a vracet booleovské návratové hodnoty podle toho, zda byla nebo byla hodnota přiřazena.

5. Vytvořte metodu addToPhase, která bude připočítávat jedničku k proměnné phase, ale tak, aby v proměnné byla vždy hodnota v rozmezí 0 až liveTime−1 (tedy 799). Místo podmínky if k tomu použijte vhodnou matematickou operaci (nápověda: když dělím číslem 4, dostávám jako zbytky po dělení 0 až 3). Vytvořte testy isGrowing (true, je-li fáze růstu), isEatable (true, je-li fáze zralosti) a isRotting (true, je-li fáze hniloby). Dodržte, že první čtvrtinu času rostlina roste (ostrá nerovnost), druhou a třetí čtvrtinu je poživatelná (neostrá nerovnost u obou hranic) a poslední čtrtinu hnije (neostrá nerovnost). Používejte striktně konstantu liveTime a její násobení či dělení, místo konkrétních čísel (aby bylo možné dodatečně změnit činnost pouhou úpravou konstanty) – v těchto metodách stačí jen return a vhodně zapsaná nerovnost. Až budete mít metodu isGrowing, využijte ji v addToPhase k doplnění kódu, který během „addToPhase“ nastaví počet mrkví na maximální možný, pokud je mrkev ve fázi růstu. Vytvořte metodu getPhaseTimePercent, která bude vracet počet procent uběhnutého času v dané fázi růstu. Uvědomte si, že je nutné počítat zvlášť (if) pro každou ze tří fází. Nalezněte přepočet, jak zařídit, že např. číslo 200 je 0 % a 600 je 100 % fáze zralosti (viz pravidla pro fáze výše).

6. Vytvořte metodu redrawImage, která bude vykreslovat nový obrázek mrkve. Používejte atribut imageBuffer, který vyplníte nejprve oranžovou barvou a potom přes něj překreslíte obrázek z imageCarrot. Nezapomeňte na závěr vhodnou metodou třídy Actor (kterou Carrot podědila) nastavit obrázek imageBuffer jako obrázek třídy. Barvy se v Greenfootu volí pomocí třídy java.awt.Color, kterou musíte naimportovat. Třída má buď konstruktor, kde lze nastavit

červenou, zelenou a modrou složku nebo statické atributy, které obsahují některé přednastavené barvy. Vizte dokumentaci: http://docs.oracle.com/javase/7/docs/api/java/awt/Color.html

7. Doplňte do redrawImage kreslení obdélníčku v horní části, který se kreslí pouze ve fázi zralosti a zobrazuje zeleným pruhem procentuální množství zbylých mrkví a pod ním žlutý (nebo oranžový) procentuální zbývající čas, po který bude možno jíst (viz obrázek). Černý obdélníček je široký 42 a vysoký 6 px a je umístěný 4 px zleva a 1 px shora. Zelený pruh je zleva 5 px a shora 2 px, široký nejvýše 40 a vysoký 2 pixely. Podobně žlutý, který je jen níž (shora 4 px).

8. Do metody act() doplňte volání dvou dříve naprogramovaných metod, které způsobí nejprve přepočítání fáze a pak překreslení obrázku. Doplňte také metodu eatOne(), která po zavolání sníží hodnotu count o 1, pouze pokud nějaká mrkev ještě zbývala a pokud je instance zrovna ve fázi zralosti (isEatable). Vracejte true nebo false, podle úspěšnosti (snědeno, nesnědeno).

9. Chybí vytvořit metodu getCarrotColor a v metodě redrawImage nastavovat barvu pro výplň právě na hodnotu, kterou bude vracet. Metoda vrací přímo instanci třídy Color (z balíku java.awt) a to tak, že použije vhodný konstruktor třídy Color, kterému předá červenou, modrou a zelenou složku barev. Ve fázi zralosti předávejte barvu s RGB hodnotami: 249, 117, 0 (oranžová). Ve fázi zrání předávejte barvu, která přechází podle procenta času (getPhaseTimePercent) od zelené (RGB hodnoty 32, 106, 35) po výše uvedenou oranžovou. Ve fázi hnití nechť barva postupně přechází od oranžové po hnědou (RGB hodnoty 93, 74, 34). Jak spočítáme odstíny barev mezi krajními barvami podle procenta času? K uložení čísel barev v metodě nepoužívejte samotné proměnné int, ale použijte vždy 3prvkové pole int pro každou barvu.