Programare orientată pe obiecte

#2 JAVA
Greenfoot (partea a I-a)

https://www.runceanu.ro/adrian/activitate-didactica/

Curs 2

GREENFOOT – mediu de programare vizuală

2. GREENFOOT

1. Concepte introductive
2. Mediul integrat de dezvoltare Greenfoot
3. Exemplu de scenariu
4. Referințe bibliografice

Concepte introductive

- O aplicatie software este o combinatie dintre un cadru de dezvoltare aplicatii in doua dimensiuni si un **IDE** (An **integrated development environment – un mediu de dezvoltare integrat**) - este o aplicatie care furnizeaza facilitati importante programatorilor pentru dezvoltarea de aplicatii.
- Un IDE contine:
  - Un editor de cod sursa (source code editor)
  - Instrumente de construire automata (build automation tools)
  - Un depanator de erori (debugger)
- **Greenfoot** este un instrument pentru invatarea limbajului JAVA
- **Greenfoot** se poate descarca de pe site-ul: http://www.greenfoot.org/

Concepte introductive

- Mediul de dezvoltare **Greenfoot** este compus din:

1. Un editor de cod sursa
2. Un browser de clase
3. Un controller pentru compilare
4. Un controller pentru executie

Concepte introductive

- Etape in proiectarea unui program **Greenfoot**:
  - Se porneste de la un scenariu existent
  - Se introduc obiecte si clase
  - Se lucreaza la interactiunea dintre clase
  - Se seteaza miscarile care au loc in lumea virtuala a aplicatiei
  - Se poate include comportament aleator al obiectelor si in plus, sunete

Concepte introductive

- La fel ca si alt mediu de dezvoltare - BlueJ, si **Greenfoot** se poate folosi pentru invatarea programarii orientate obiect intr-un maniera vizuala.

- Fiecare actor este un obiect care se misca in interiorul lumii (care este de asemenea un obiect).

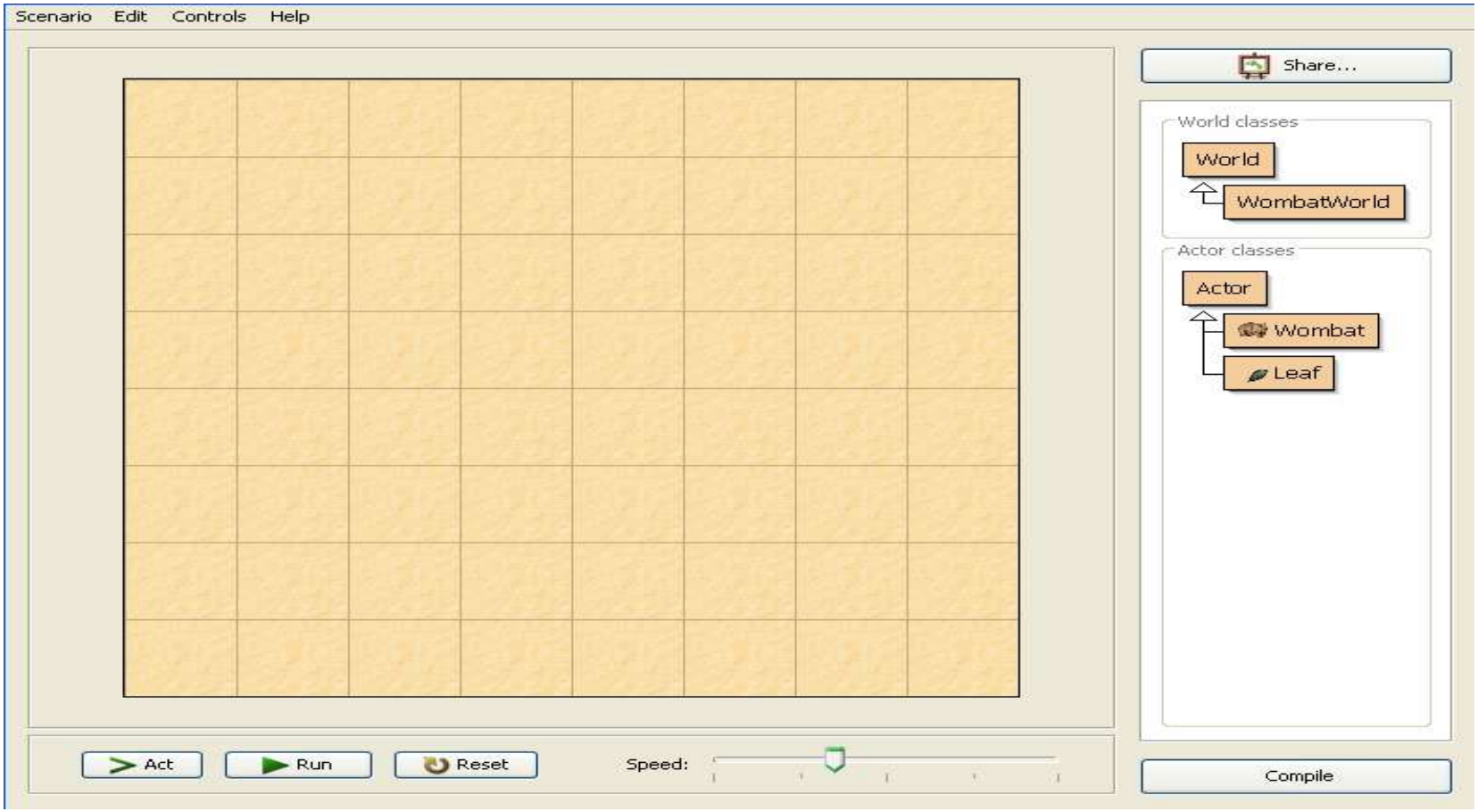
- Aceasta abordare permite invatarea principiilor programarii orientate obiect (apelarea metodelor, starea obiectelor), chiar inainte de a incepe sa se citeasca codul sau inainte de a scrie cod.

2. GREENFOOT

1. Concepte introductive
2. Mediul integrat de dezvoltare Greenfoot
3. Exemplu de scenariu
4. Referințe bibliografice

Greenfoot IDE

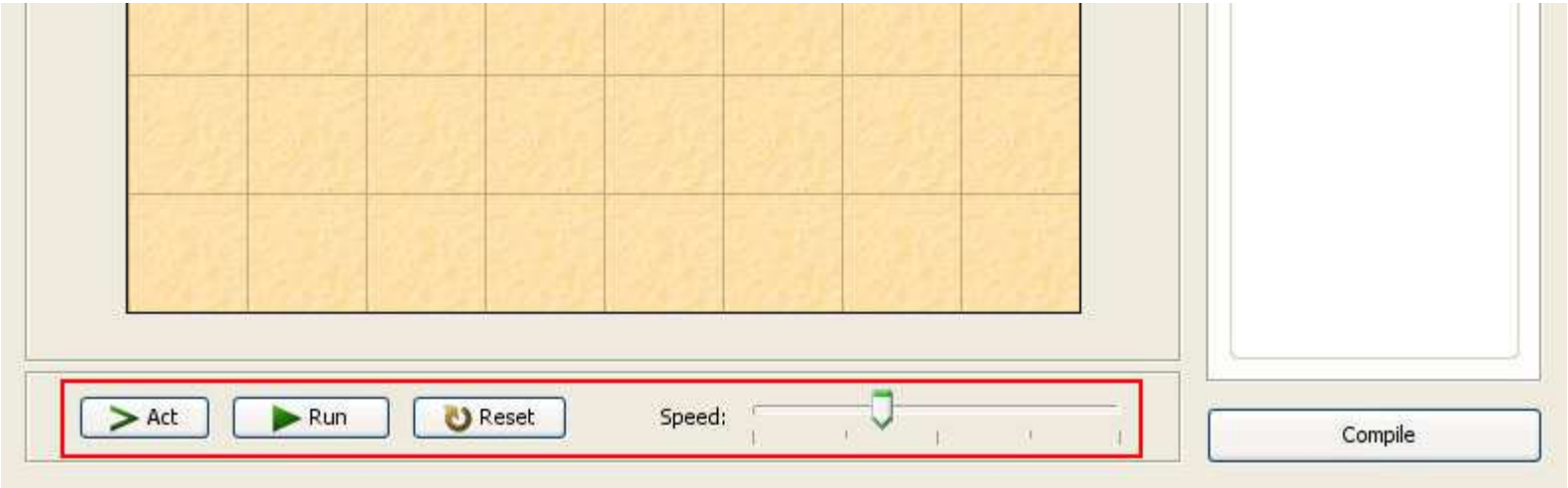
Mediul de dezvoltare **Greenfoot** IDE arata astfel:



Executia unui program

Optiunile ce controleaza executia unui program sunt:

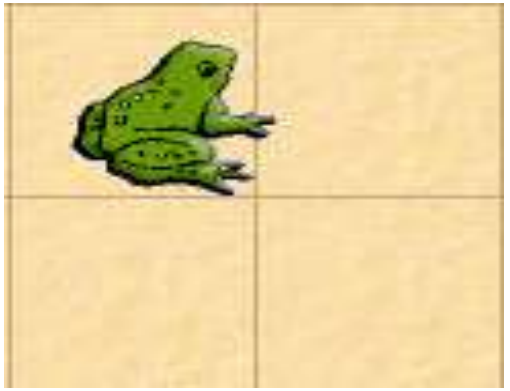
- **Act**: Executa toate actiunile din scenariu o singura data
- **Run**: Executa toate actiunile din scenariu in mod repetat pana cand apasam **Pause**
- **Reset**: Opreste scenariul sau il reseteaza inapoi la starea initiala
- **Speed**: Permite modificarea vitezei



Exemple de clase

Există mii de broaște din lume, care toate împărtășesc unele caracteristici comune:

- doi ochi
- patru picioare
- piele lipicioasa
- și capacitatea de a face un sunet "Oac"



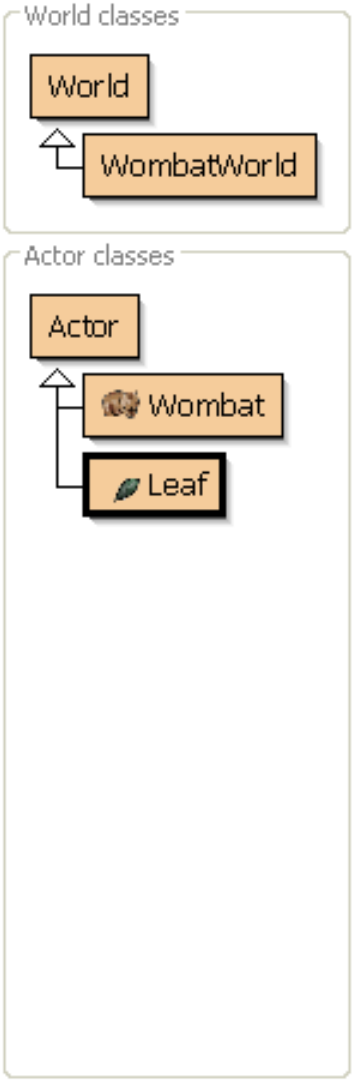
- Pădurea in care o broască trăiește poate contine arțari.
- Există sute de astfel de copaci, care au coaja similara, frunze, și au aceleași dimensiuni și culori.

Clase

- În programare, clasa ne ofera informații generale despre modul in care obiectul arată, se mișcă și ce poate face.

O clasă este compusa din atribute generale care definesc un obiect, cum ar fi aspectul, caracteristici și mișcare.

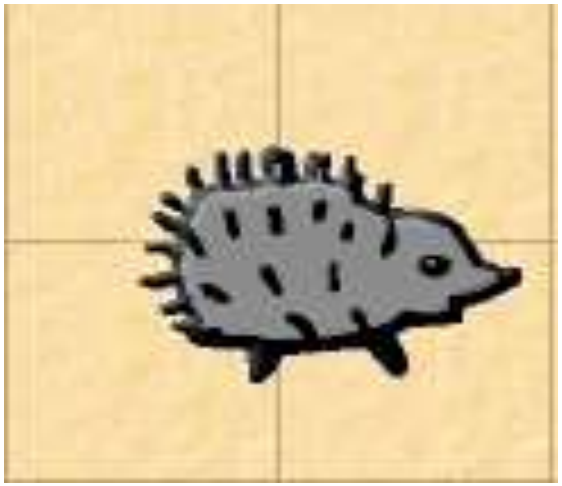
O instanta este un obiect al clasei.



- Prin adăugarea unei clase la scenariul existent, aceasta vă oferă posibilitatea de a pune acele tipuri de obiecte - sau instanțe ale clasei - în scenariu

Subclase si superclase

- Există mai multe specii de arici.
- Fiecare specie are proprietățile generale ale clasei arici, cum ar fi pielea cu ace, un nas ascuțit și patru picioare, dar si diferențe de culoare, formă, mărime și care sunt specifice speciei.



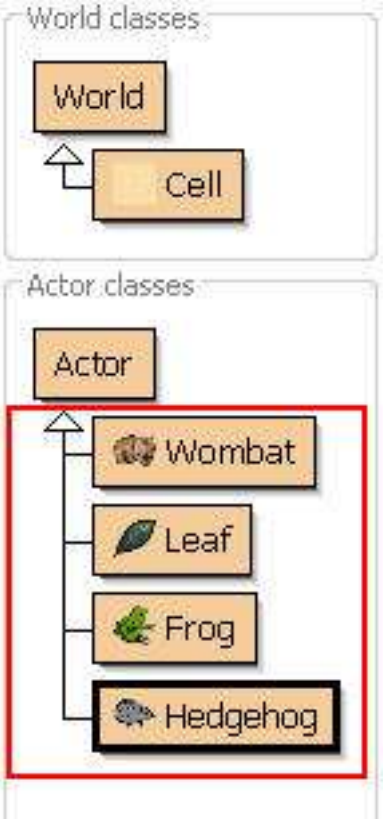
Clasa generala dintr-un grup de clase se numeste superclasa.

Clasele derivate dintr-o clasa se numesc subclase.

Proprietatile subclasei

O subclasa:

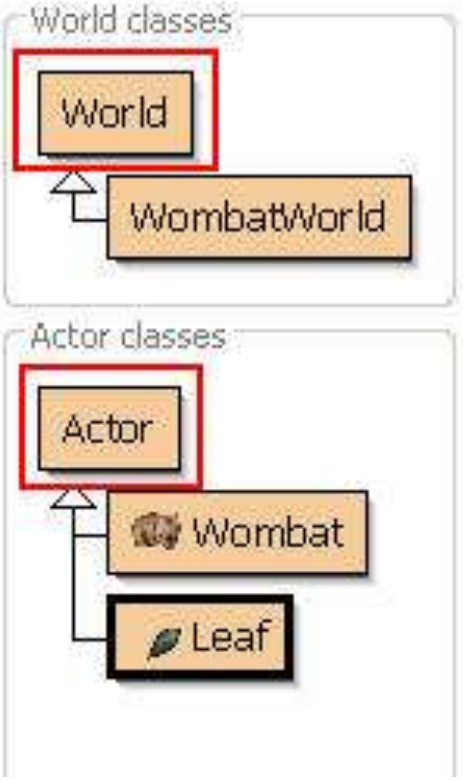
- Este o clasa aflata in relatia “este-o” subclasa a superclasei (adica, clasa arici este o subclasa a superclasei actor)
- Are propriile instante
- Poate fi redenumita
- Contine o sageata care arata din ce superclasa provine in ierarhia de clase



Tipuri de superclase

Cele două tipuri de superclasa utilizate în Greenfoot sunt:

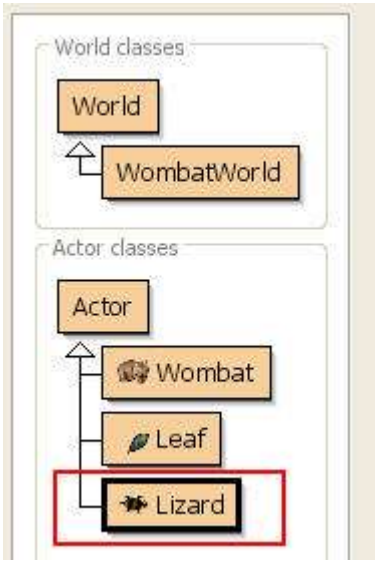
- **World**:
  - gestioneaza toate subclasele aflate in lumea virtuala
  - Definește dimensiunea și rezoluția din lumea virtuala
- **Actor**:
  - gestioneaza subclasele care produc instante de tipul actor



Pasi pentru creare unei subclase

Pentru a crea o subclasă:

1. Faceți clic dreapta pe o superclasa.
2. Selectați New subclass ...
3. Alegeti un nume clasei.
4. Selectați o imagine de clasă din meniu, apoi faceți clic pe OK.
5. Subclasa apare în meniul clasei.



Crearea unei subclase si importarea unei imagini

Pentru a importa o nouă imagine de pe computer in vederea atribuirii acesteia noii subclase:

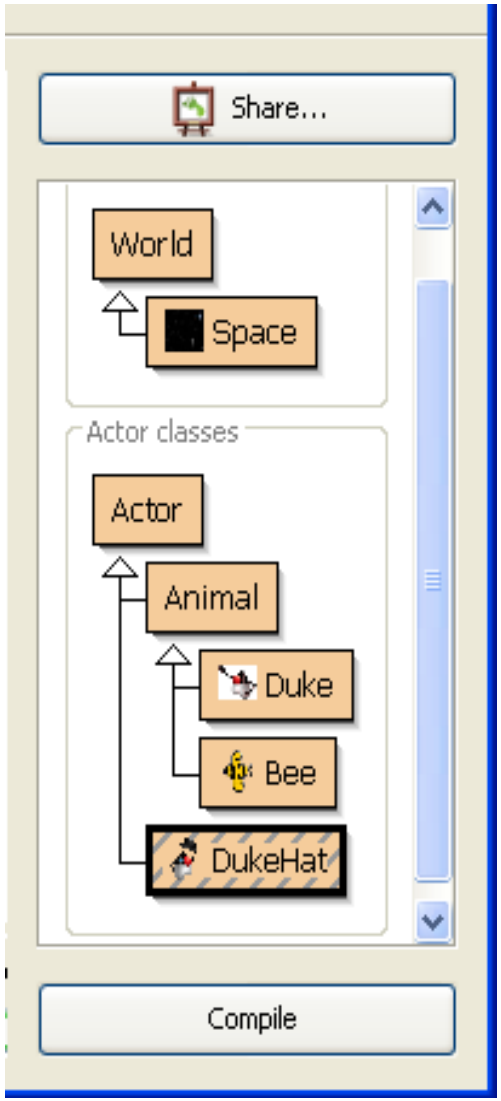
1. Faceți clic dreapta pe o superclasa.
2. Selectați New Subclass ...
3. În fereastra New Class, faceți clic pe butonul Import
4. Apasati pe butonul File ...
5. Selectați fișierul pentru a-l importa din computer.
6. Denumiti noua subclasa, apoi faceți clic pe OK.
7. Subclasa cu noua imagine apare în meniul clasei.

Compilarea

Compilarea se executa dupa ce subclasa a fost definita.

Compilarea traduce codul sursă în cod mașină pe care computerul il poate înțelege.
Aspectul hasurat specifica faptul că ați adăugat cod sursă sau de clasă în mod corect înainte de a continua

După ce adăugați o clasă la un program, sau dupa ce scrieti cod sursă care instruiește clasa cu privire la modul de a acționa, nu puteți adăuga clase suplimentare sau cod sursă până când programul este compilat.



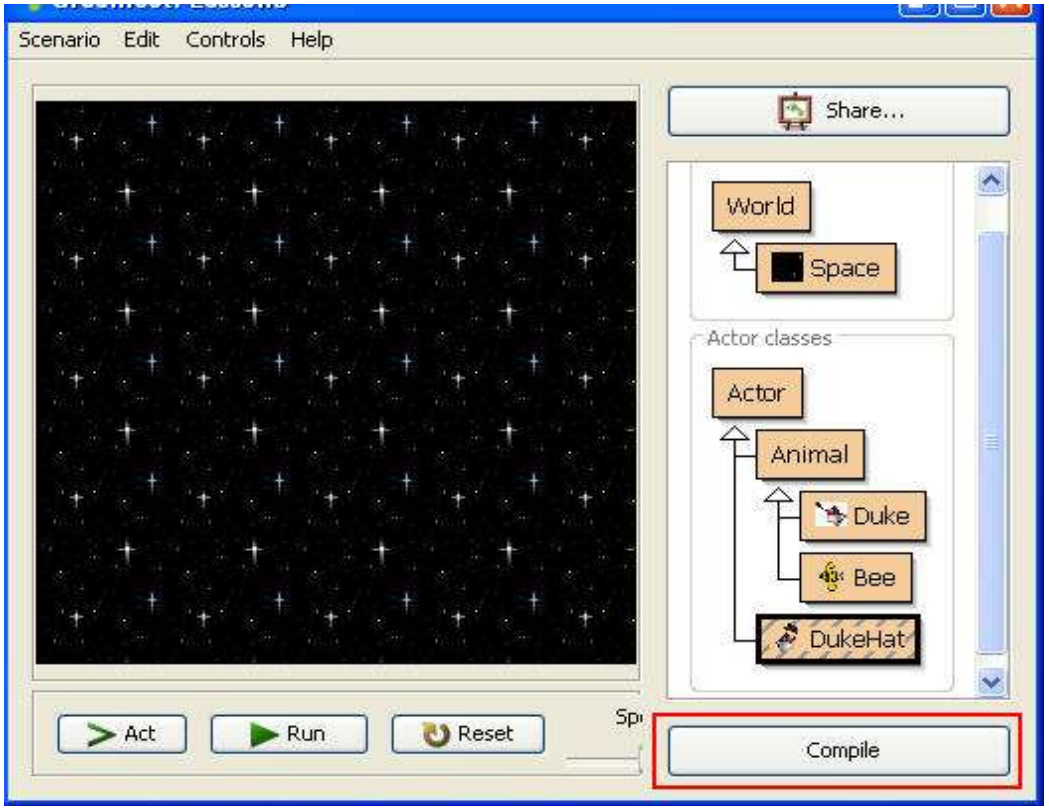
Trebuie efectuata compilarea

Compilarea trebuie efectuata cand:

- Codul sursa este modificat
- Inainte ca noi instante ale clasei sa fie create
- Pentru toate subclasele, daca superclasa din care provin, a fost creata sau a fost modificata

Compilarea unui program

- Apasa **Compile**. Odata operatia de compilarea efectuata, hasurile dispar.
- Apoi se continua partea de programare sau se executa scenariul (**Run**).



Salvarea unui scenariu

Pentru a salva un scenariu se urmeaza pasii urmatori:

1. In meniul Scenario, se selecteaza Save a Copy As...
2. Save a copy intr-un folder aflat pe calculatorul vostru.

De fiecare data cand inchideti **Greenfoot**, se salveaza in mod automat ceea ce ati modificat.

Se pot salva versiuni multiple ale scenariilor, pentru:

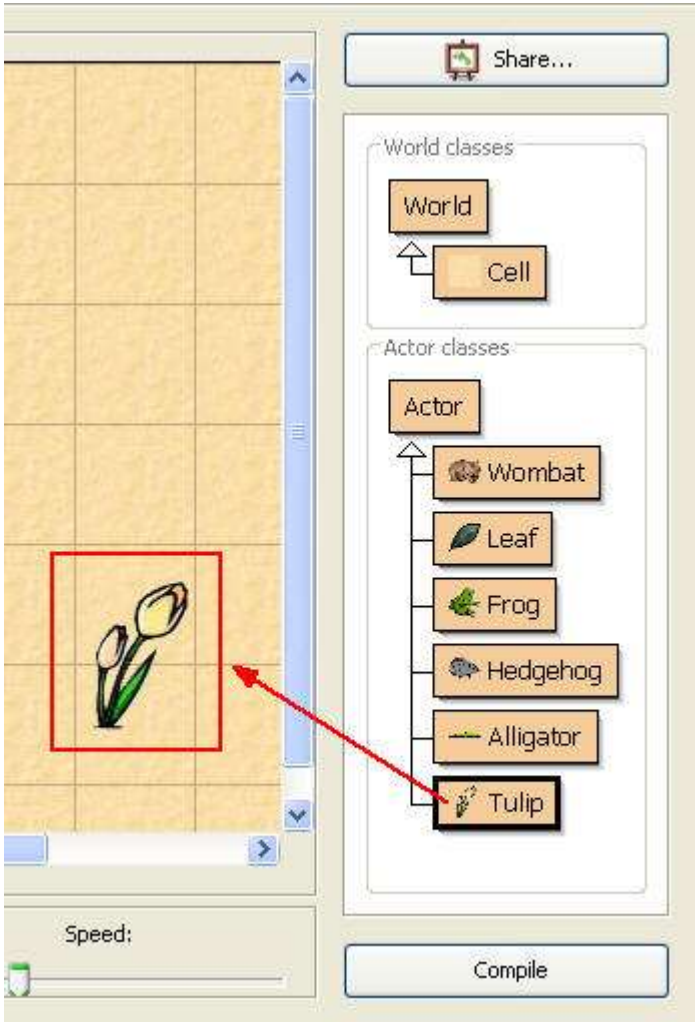
- A va putea intoarce la versiuni anterioare ale unui scenariu
- A putea avea mai multe scenarii la care sa lucrati

Instante

- Puteti adauga oricate instante ale unei clase doriti intr-un scenariu.

Instantele sunt obiecte ale unei clase care actioneaza in scenariul vostru.

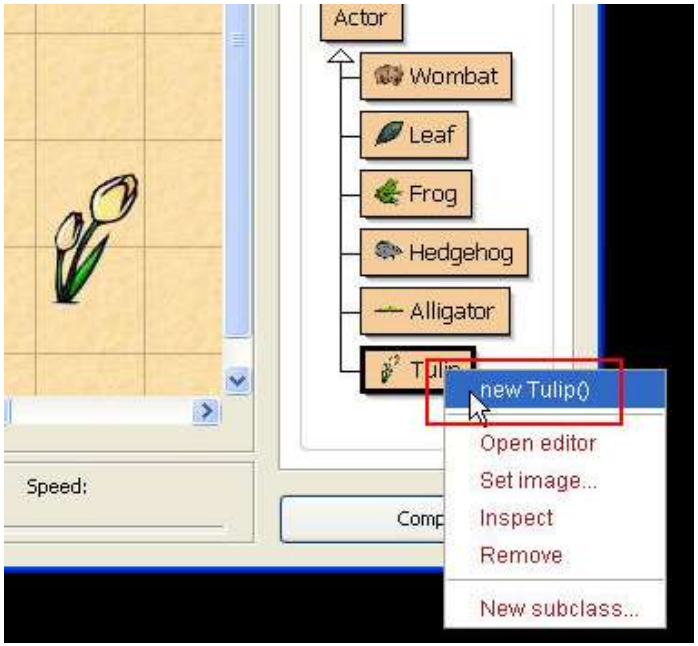
- Instantele pot executa comportamentele care sunt detaliate in codul sursa al clasei.

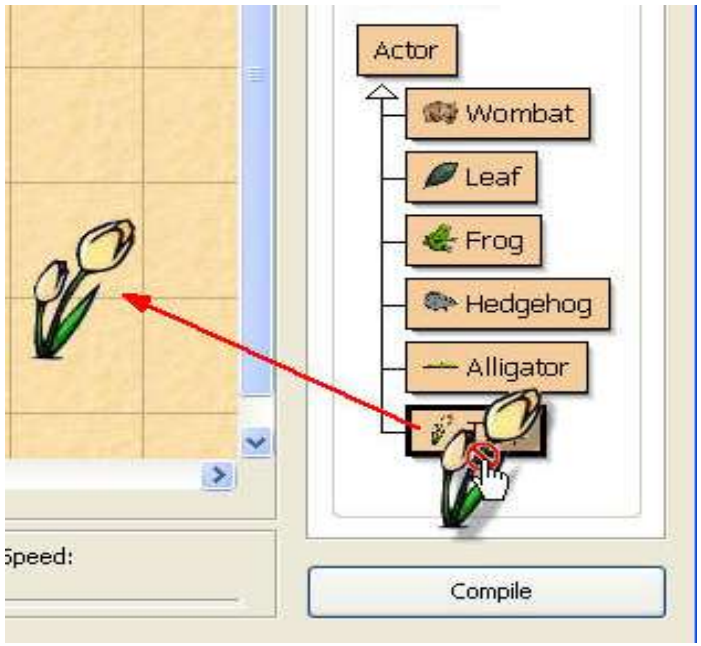


Adaugarea unei instante la un scenariu

Pentru a adauga o instanta la un scenariu se executa urmatorii pasi:

1. Right click pe clasa.
2. Click pe optiunea new[class name].
3. Trage(Drag) instanta in scenariu cu ajutorul mouse-ului.
4. Programeaza noua instanta in scenariul respectiv.

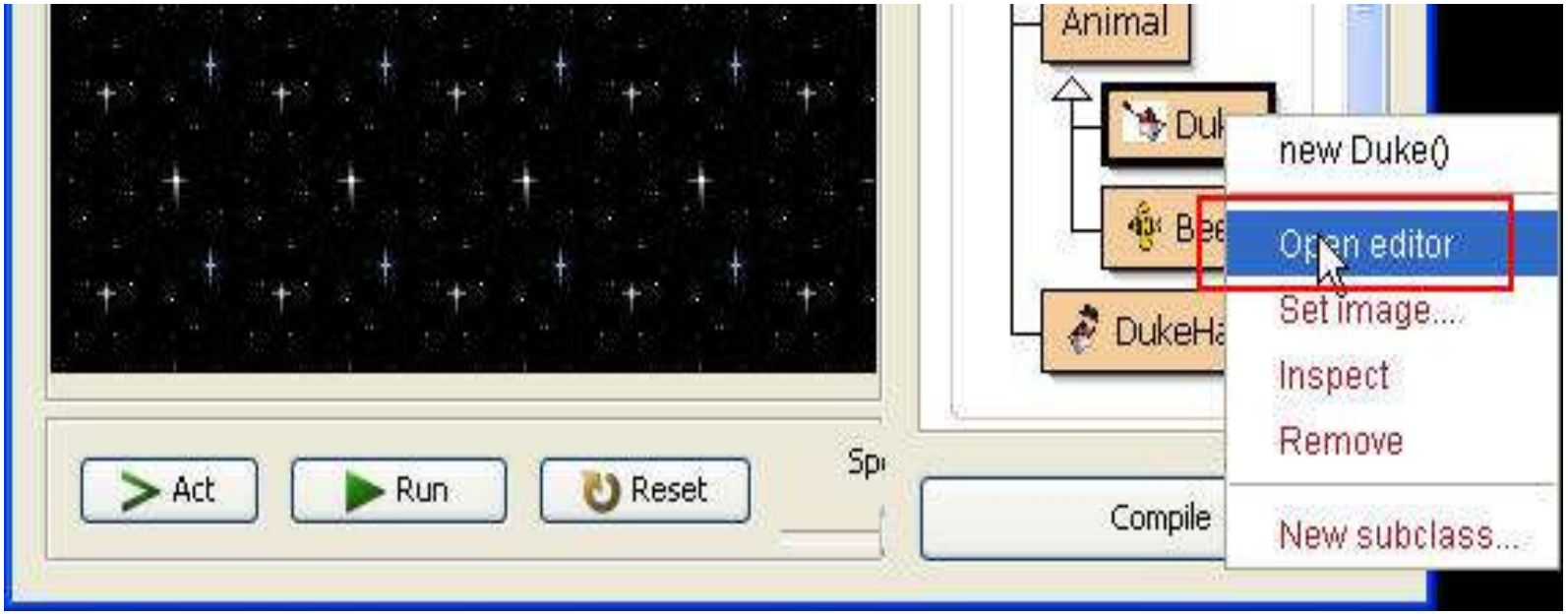




Vizualizarea codului sursa

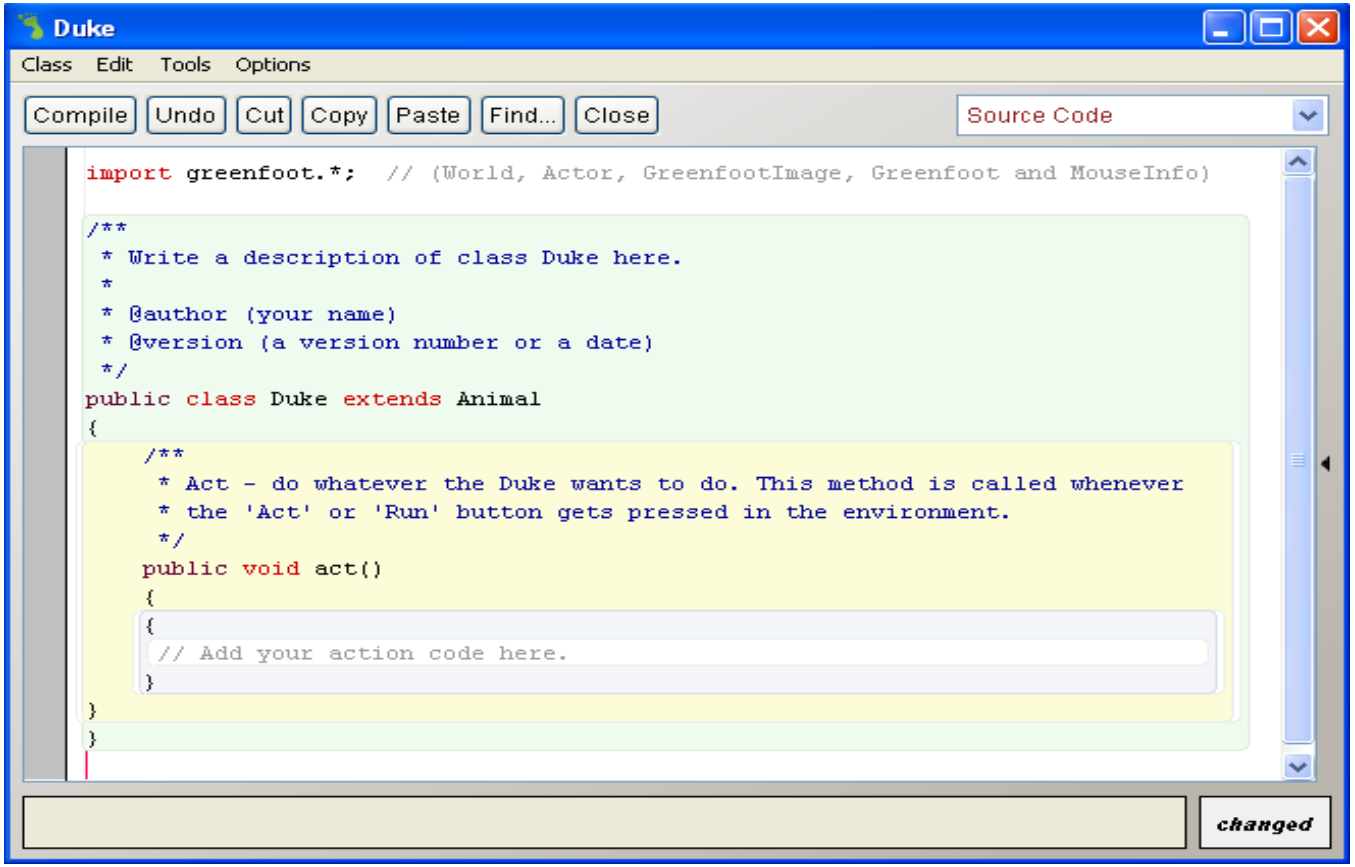
Pentru a vizualiza codul sursa al unei clase:

1. Se apasa butonul Right pe clasa.
2. Se selecteaza Open Editor.



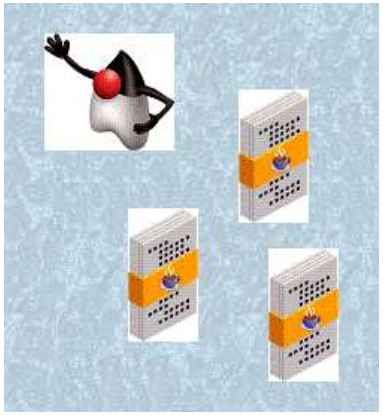
Editorul de cod

- Editorul de cod afiseaza codul sursa al oricarei clase din scenariu.
- Aici este locul unde se scriu instructiuni pentru toate instantele care sunt programate.



Crearea automata de instante ale clasei Actor

- Puteți scrie cod în constructorul World care va adăuga instanțele Actor la joc, atunci când scenariul este inițializat.
- Aceasta elimină necesitatea ca jucătorul sa adauge manual instante înainte de a începe jocul.



- De exemplu, in jocul de potrivire a cartilor de joc, acestea trebuie afisate automat in scenariu, atunci cand jocul se porneste.

Codul pentru crearea automata de instante

Codul sursa in constructorul World include urmatoarele componente:

- Instructiunea `super()`
- Dimensiunea lumii virtuale (World) ca argument in metoda `super()`
- Metoda `addObject`
- Cuvantul cheie `new` pentru adaugarea unui nou obiect
- Localizare obiectului cu metoda `addObject`

```
public DukeWorld()
{
      super(560, 560, 1);
      addObject (new Duke(), 150, 100);
}
```

Constructor obtinerea unei noi imagini a unui obiect

Creați un constructor care primeste un nou obiect imagine dintr-un fișier atunci când se creează o instanță a unei clase.

Acest constructor ar putea include:

- O metoda `SetImage new`
- Cuvantul cheie al clasei `GreenfootImage` class
- Numele fisierului ca argumente in lista de parametri

Constructorul de mai jos creează noua imagine, si-o ataseaza clasei Actor.

```
setImage (new GreenfootImage("duke5.png"));
```

Asignarea unei noi imagini la o clasa

- Declarația de mai jos creează noul obiect imagine din numele fișierului imagine specificat.
- Când se introduce în codul sursă al clasei, acest obiect imagine este gata pentru a fi utilizat.

Instructiunea care se executa astfel:

- Obiectul `GreenfootImage` este creat la inceput.
- Metoda `setImage` este executata, transmitand obiectul de tip imagine nou creat ca argument in lista de parametri.

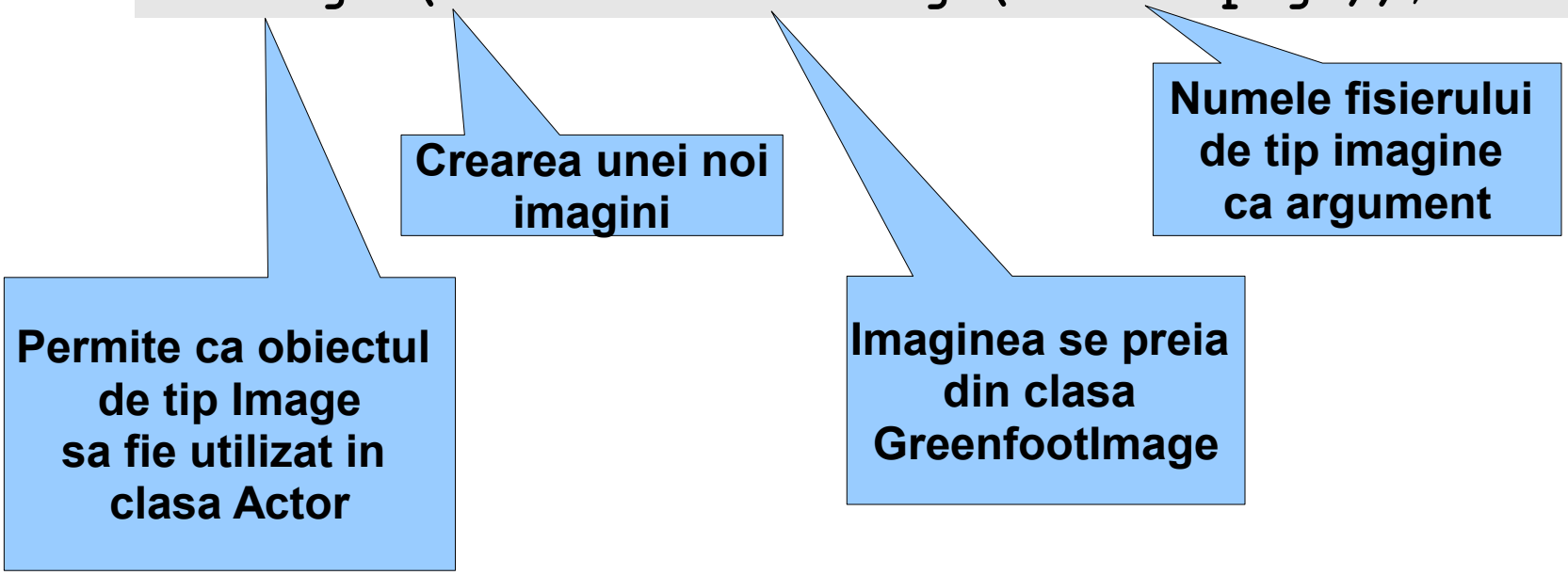
```
setImage(new GreenfootImage ("duke5.png"));
```

Asignarea unui exemplu de tip Image

Metoda `setImage` seteaza imaginea din fisierul “duke5.png” in clasa Actor.

De fiecare timp cand o instanta a acestei clase este adaugata in scenariu, se afiseaza imaginea din fisierul “duke5.png”.

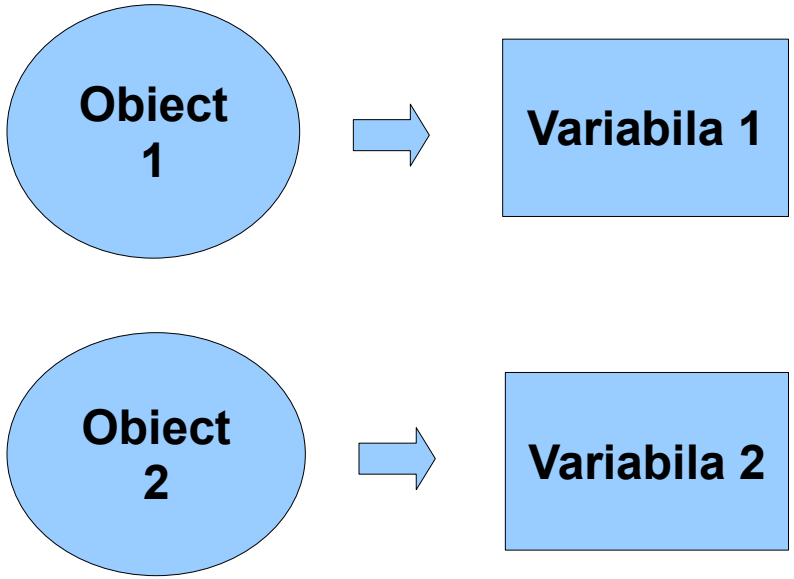
```
setImage (new GreenfootImage("duke5.png"));
```



Variabile

Se pot utiliza variabile pentru a stoca doua obiecte de tip imagine in clasa, si astfel clasa sa poata accesa mai usor instantele lor.

O variabila este declarata intr-o clasa. Ea se utilizeaza pentru a stoca informatii ce vor fi utilizate ulterior. In variabila se pot stoca obiecte sau valori.



Formatul variabilelor

Formatul variabilelor contine:

- Tipul de date: Ce tip de date este stocat in variabila respectiva.
- Numele variabilei: Descrie identificatorul variabilei care va fi folosita ulterior in program.

```
private tip-variabila  nume-variabila;
```

Exemplu:

Numele variabilei este image1 iar tipul variabilei este GreenfootImage.

```
private GreenfootImage image1
```

Declararea variabilelor

Declararea variabilelor se face înainte de declararea constructorilor și a metodelor.

Formatul pentru declararea unei variabile, include:

- Cuvantul cheie **private** pentru a indica faptul că variabila este disponibila numai în clasa Actor.
- Clasa din care face parte imaginea.
- Inlocuitor pentru variabila în care va fi stocata imaginea.

```
/**
 * Write a description of class Duke here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Duke extends Animal
{
private GreenfootImage image1;
private GreenfootImage image2;
```

Variabile

Instructiunea de atribuire

- O atribuire este necesara pentru a stoca obiecte in variabile.
- Cand un obiect este atribuit unei variabile, variabila va contine o referinta la acel obiect.

O instructiune de atribuire:

- Stocheaza un obiect sau o valoare intr-o variabila
- Se utilizeaza simbolul "="

Format

```
Variabila = expresie;
```

Componentele instructiunii de atribuire

O instructiune de atribuire include:

- ✓ Variabila: Numele variabilei care va stoca un obiect sau o valoare
- ✓ Simbolul "="
- ✓ Expresie:
  - Numele unui obiect sau o valoare care fi setata
  - O instructiune prin care un obiect nou este creat sau o valoare este adaugata
  - Clasa careia apartine imaginea respectiva

Exemplu:

```
image1 = new GreenfootImage("duke.png");
image2 = new GreenfootImage("duke2.png");
```

Exemplu de constructor al clasei Actor

Urmatorul constructor al clasei Actor transmite mediului Greenfoot sa creeze automat o noua instanta de tip Duke care sa fie initializata sau sa i se atribuie o valoare.

Se creeaza automat doua instante.

```
public class Duke extends Animal
{
    private GreenfootImage image1;
    private GreenfootImage image2;
    /**
     * Create a Duke and initialize his two images.
     */
    public Duke()
    {
        image1 = new GreenfootImage("duke.png");
        image2 = new GreenfootImage("duke2.png");
        setImage(image1);
    }
```

Ultima linie de cod din constructor apeleaza metoda setImage(image1), care precizeaza ca prima variabila trebuie afisata cand instanta respectiva este adaugata in scenariu.

Sfarsitul jocului

- Clasa **Greenfoot** contine o metoda Stop care poate fi folosita pentru a termina un joc la momentul in care programatorul stabileste aceasta.

Aplicatia(jocul) poate fi terminata cand:

- ✓ Jucătorul atinge o piatră de hotar (margine)
- ✓ Expira timpul de joc.
- ✓ Instanța atinge anumite coordonate sau obiecte.

Folosirea metodei Stop in aplicatie(joc)

- ✓ Atunci cand se doreste incheierea jocului, atunci de apeleaza metoda de oprire.

```
Greenfoot.stop();
```

Exemplu de control a tastaturii in metoda Act

- ✓ In metoda Act, controlul tastaturii se poate defini cu metoda lookForCode.

```
public class Duke extends Animal
{
    /**
     * Act - do whatever the Duke wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        move(3);
        if (Greenfoot.isKeyDown("left"))
        {
            turn(-2);
        }
        if (Greenfoot.isKeyDown("right"))
        {
            turn(2);
        }
        lookForCode();
    }
```

2. GREENFOOT

1. Concepte introductive
2. Mediul integrat de dezvoltare Greenfoot
3. Exemplu de scenariu
4. Referințe bibliografice

Exemplu de scenariu pentru construirea unui joc* amuzant!

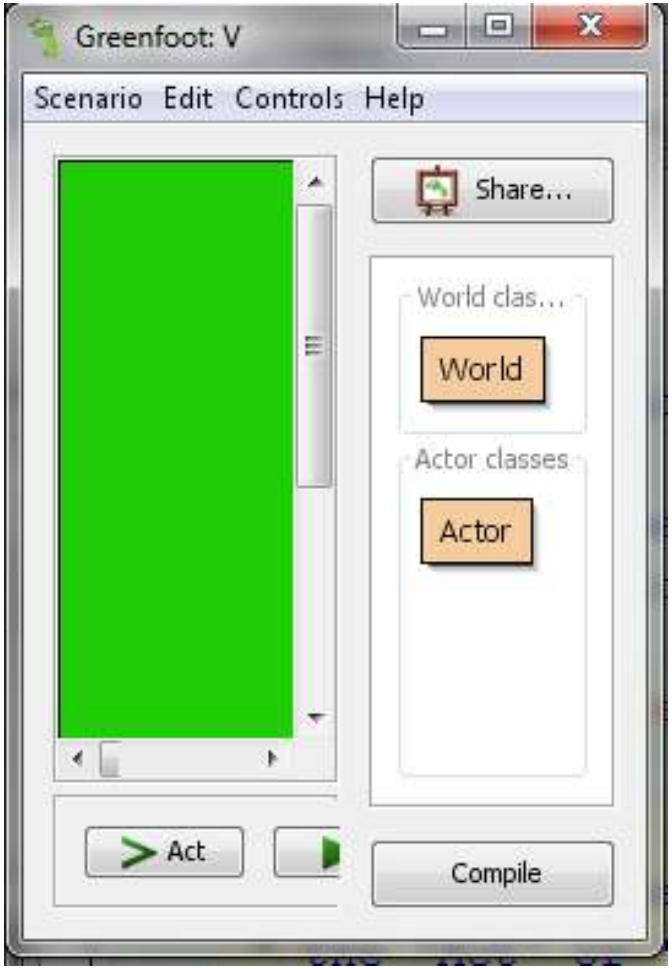
*Idee implementata de prof. Dana Vaida

Scenariul

- Vasile ursuletul, iubeste mierea si se decide sa adune albine pentru a-si forma propriul stup.
- Nu are cunostinte de apicultura si se gandeste sa stranga suficient de multe albine pe care sa le „cazeze” intr-un stup.

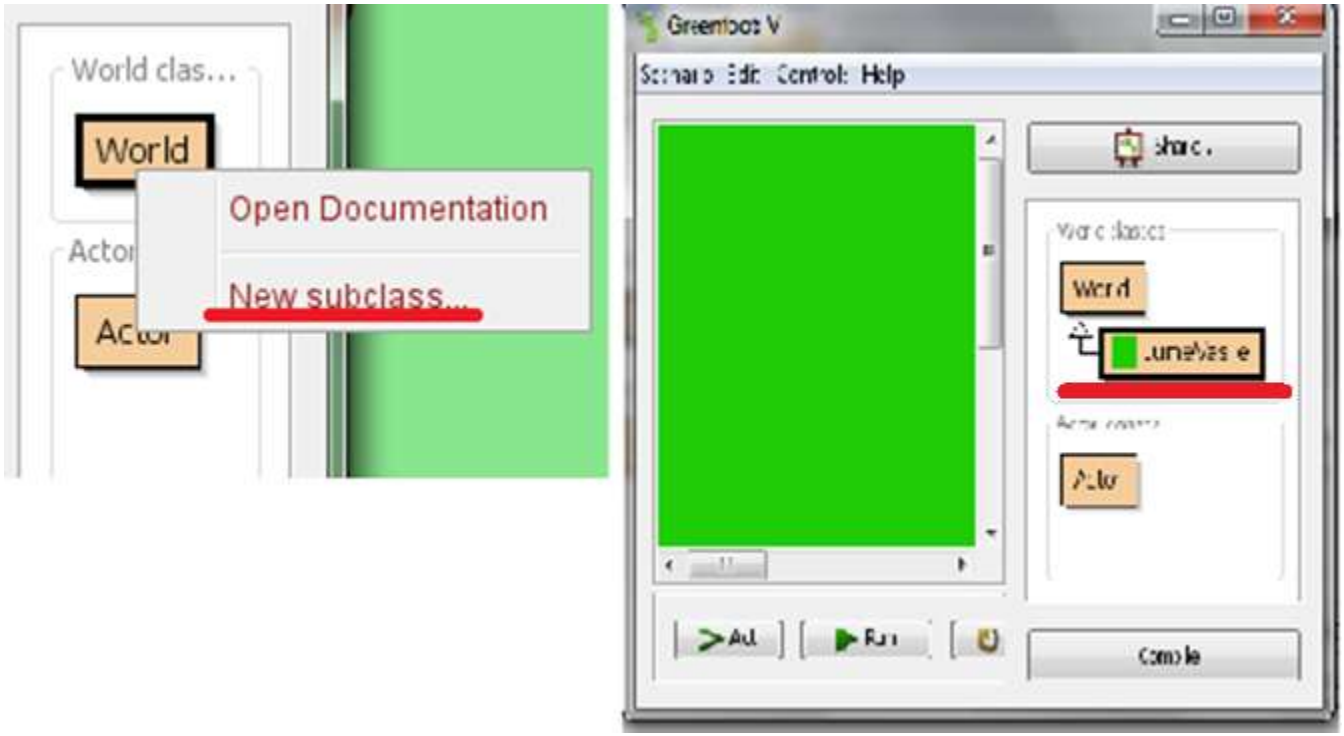
Pornirea aplicatiei Greenfoot

- Deschidem mediul de programare **Greenfoot**.
- In dreapta observam doua superclase: WORLD si ACTOR.
- Intai cream o subclasa a clasei WORLD care va reprezenta lumea in care se desfasoara actiunea.
- Apoi cream subclase ale clasei ACTOR pentru fiecare categorie de actori implicati in scenariu.



Crearea unei subclase in Greenfoot

- Vom crea clasa LumeVasile dand clic dreapta pe clasa World.



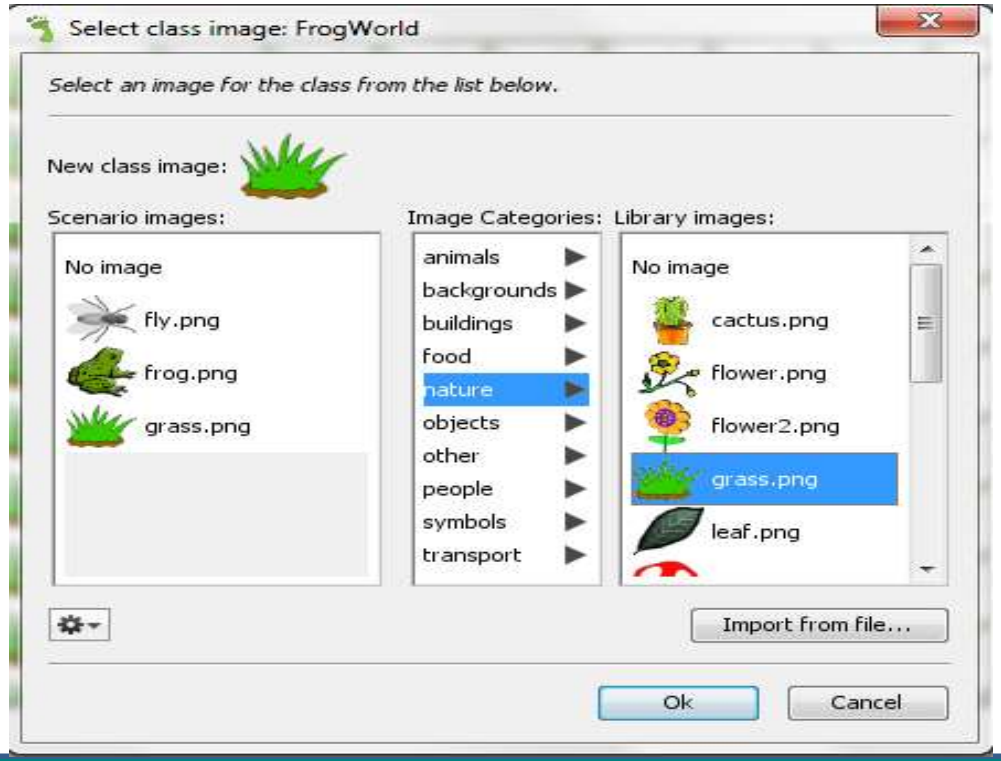
Setarea imaginii unei clase - manual

1. Atunci cand creezi clasa:

- **alege o imagine din lista**,
- sau **deseneaza** una
- sau **importa o imagine de pe calculator.**

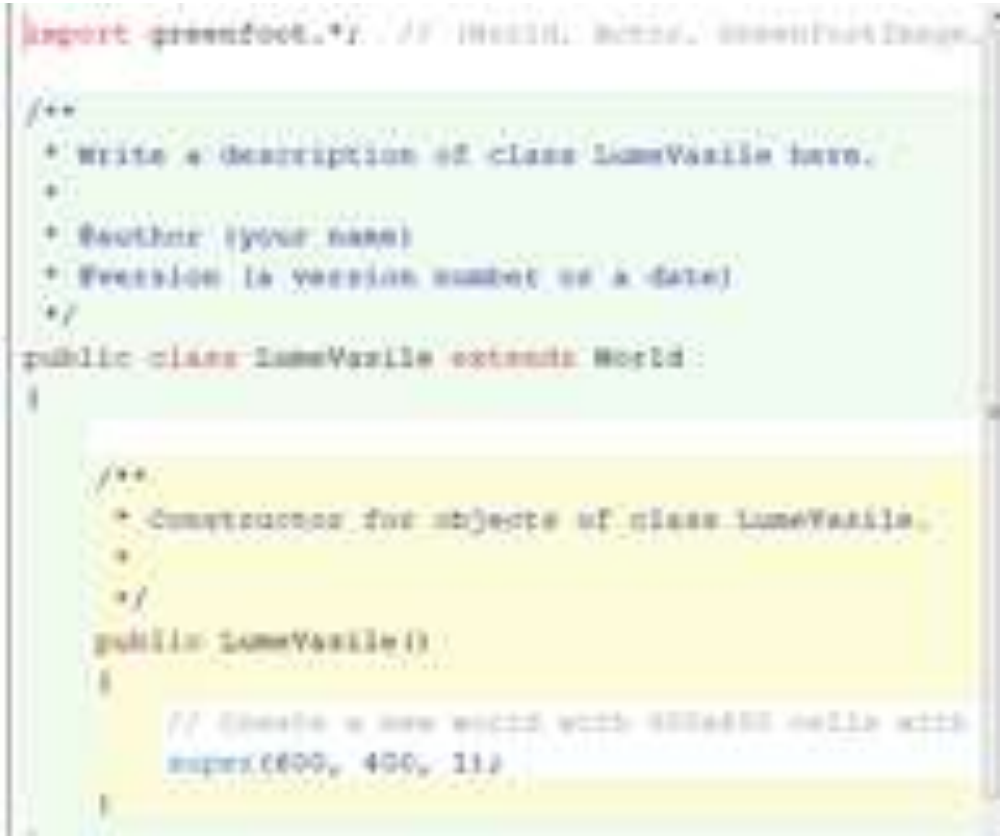
Setarea imaginii unei clase - manual

2. Se poate face si dupa creare, alegand optiunea Set Image. Se deschide fereastra de mai jos, unde putem alege o imagine din lista, putem desena una sau putem importa o imagine de pe calculator.



Sursa corespunzatoare crearii clasei

- Am creat clasa LumeaVasile ca subclasa a lumii World.



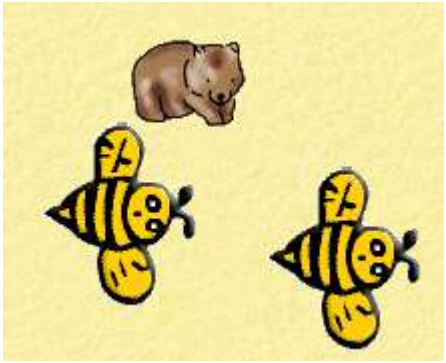
```
import greenfoot.*;  // (World, Actor, GreenfootImage,

/**
 * Write a description of class LumeVasile here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class LumeVasile extends World
{

    /**
     * Constructor for objects of class LumeVasile.
     *
     */
    public LumeVasile()
    {
        // Create a new world with 600x400 cells with
        super(600, 400, 1);
    }
```

super(600, 400, 1);

- Construieste un spatiu (lume) de 600 x 400 pixeli avand rezolutia de 1 pixel

Crearea subclaselor clasei Actor

- Cream clasa Albina.
- Daca dorim pentru albina o alta imagine decat cea gasita in mediu, putem alege o imagine pregatita in prealabil pe calculator.



Setarea Imaginii unei clase – prin programare pas 1

Intai trebuie sa declar o variabila de tip imagine.

Exemplu

```
private GreenfootImage imagine1
```

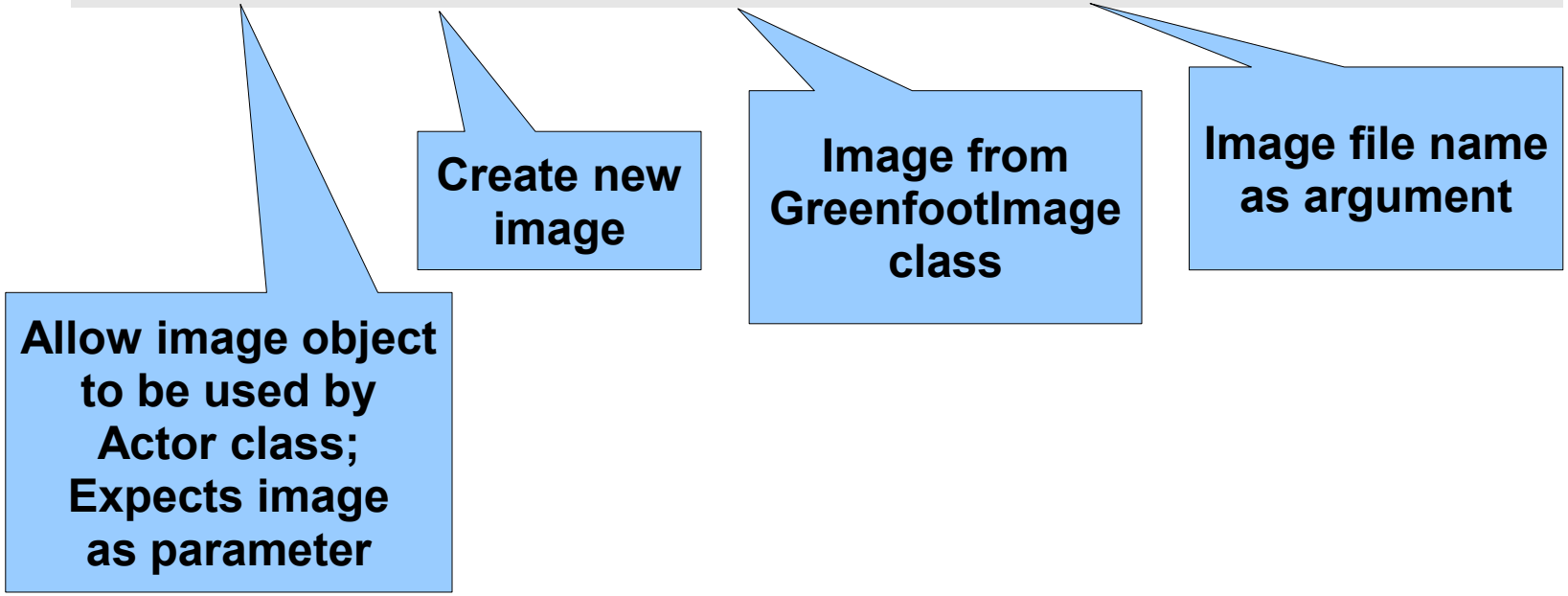
Imagine1 este o variabila de tip GreenfootImage.

Setarea Imaginii unei clase – prin programare pas 2

Metoda setImage ataseaza imaginea “albina2.png” pentru obiectele din clasa Albina.

De cate ori se va adauga o noua instanta a acestei clase aceasta va avea imaginea “albina2.png”.

```
setImage (new GreenfootImage("albina2.png"));
```



Variabile – private, public, static

Orice declarare de variabila este de forma:

```
Private/public [static]  variable_type  variable-name;
```

Unde variable_type este tipul de data pe care il va avea variabile_name.

Private – spune ca va fi „vazuta” doar in cadrul clasei in care este definita.

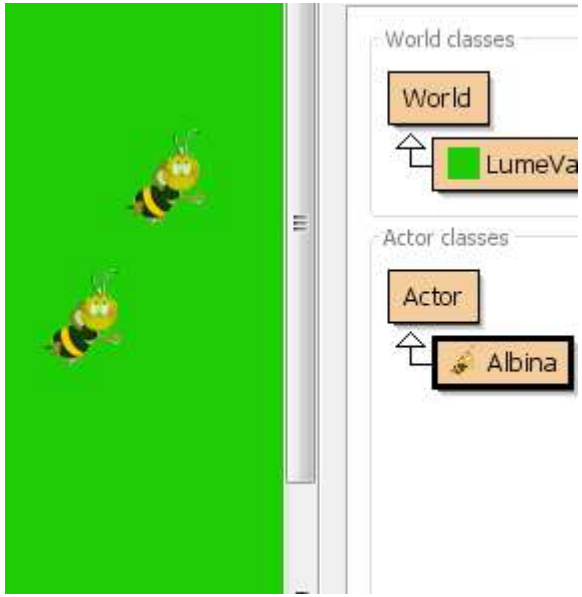
Public – va fi „vazuta” din orice clasa si se asociaza fiecarei instante deci are valori diferite pentru fiecare instanta.

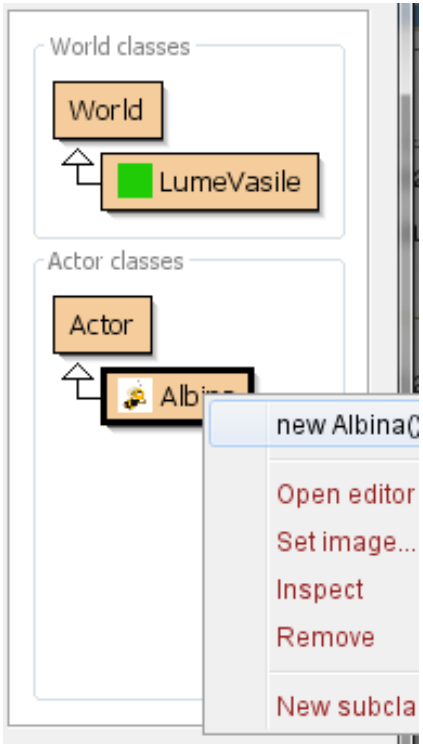
OBSERVATIE:

Daca adaug cuvantul static, variabila va fi asociata clasei si va avea o unica valoare (exemplu de utilizare: contor global)

Adaugarea manuala de instante

- O lume va fi vida daca nu adaugam niste actori in ea.
- Sa adaugam manual doua albine:
  - Clic dreapta pe actor, apoi alegem **new Albina()**.
  - Apare o albina pe care o tragem cu mouse-ul pe ecran.





Adaugare de instante

Exista trei modalitati prin care putem adauga in „Lume” (World) instante din clasele Actor.

1. Manual, chiar si atunci cand scenariul se executa.

- ✓ Avantaj: Incurajeaza utilizatorul sa adauge instante unde doreste.
- ✓ Dezavantaj: La compilare sau reset obiectele create dispar.

2. Automat, prin programare atunci cand construim lumea (in functia constructor a lumii jocului).

3. Automat, prin programare in timpul desfasurarii jocului.

Adaugarea unui obiect

- Editam codul clasei LumeVasile.
- Pentru a adauga 3 albine apelam metoda **addObject** aflata in clasa World, in functia constructor LumeVasile:

```
public LumeVasile()
  {
     super(600, 400, 1);
     addObject(new Albina(),30,200);
     addObject(new Albina(),350,100);
     addObject(new Albina(),350,300)
;…
```

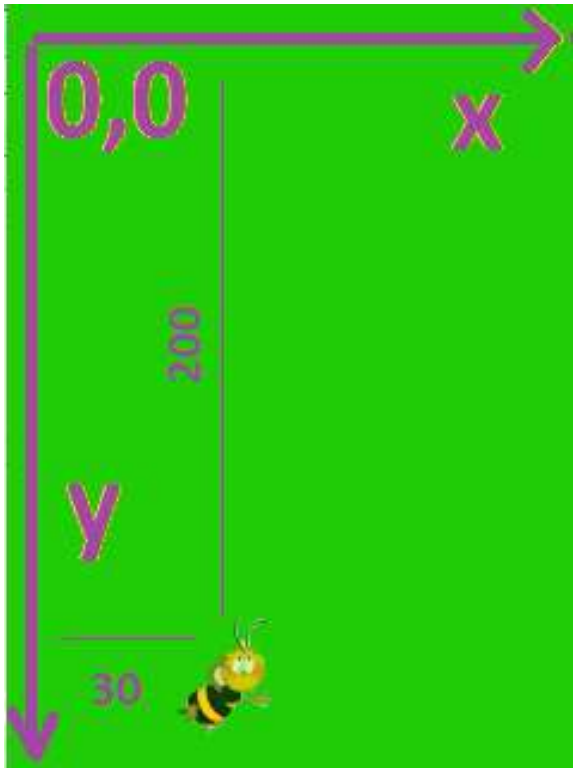
Adaugarea unui obiect

De exemplu, pentru apelul:

addObject(new Albina(),30,200);

ECRAN

Se creaza o albina ce va fi afisata pe ecran in sistem xOy avand x=30 si y=200



Adaugarea unui obiect in timpul executiei

- Daca dorim adaugarea unei albine

in timpul executiei, va trebui sa prefixam apelul astfel:

```
getWorld().addObject(new Albina(), 100, 100);
```

Animatie

A venit timpul sa animam aceste albine.

Pentru aceasta folosim metodele:

```
move(5);     // Muta obiectul pe directia curenta cu 5 pixeli
turn(3);     // Intoarce obiectul cu 3 grade
```

- Compileaza acum si executa programul, albinele se vor misca acum.
- Observam ca se misca deodata pe aceeasi traiectorie.

Animatie aleatoare

- Pentru a genera numere aleatoare apelam metoda Greenfoot.getRandomNumber(X) care genereaza un numar aleator cuprins intre 0 si X-1.

- Daca dorim ca fiecare albina sa aiba miscarea sa proprie putem inlocui parametri cu numere generate aleator astfel:

```
move(Greenfoot.getRandomNumber(3));
turn(Greenfoot.getRandomNumber(5);
```

Animatie aleatoare

- Observam ca se misca oarecum diferit dar totusi se misca intotdeauna in acelasi sens.
- Pentru ca sa se produca si schimbari de directie putem folosi o scadere, de exemplu:

```
move(Greenfoot.getRandomNumber(10));
turn(Greenfoot.getRandomNumber(40)-20);
```

- Am crescut la 10 pixeli fiecare deplasare pentru a fi mai evidenta.

Sa cream un numar aleator de albine

Sa generam acum cel mult 30 de albine asezate aleator in spatiul creat:

```
for(int i=1; i<=Greenfoot.getRandomNumber(60); i++)
        addObject (new Albina(),

Greenfoot.getRandomNumber(600),
Greenfoot.getRandomNumber(400));
```

Sa adaugam acum ursul

Pozitie fixa in World:

```
addObject (new Urs(), 500, 300);
```

Sau in pozitie aleatoare:

```
addObject (new Urs(), Greenfoot.getRandomNumber(600),
Greenfoot.getRandomNumber(400));
```

Controlul obiectului prin intermediul tastaturii

In metoda Act a clasei Urs adaugati urmatoarea secventa de cod:

```
public void act()
   {
     move(3);
     if (Greenfoot.isKeyDown("left"))  turn(-2);
     if (Greenfoot.isKeyDown("left"))  turn(2);  …
```

Complilati si executati.
„left” corespunde sagetii <- iar „right” este ->

Controlul obiectului prin intermediul tastaturii

- Pentru a deplasa ursul pe cele patru directii cu ajutorul sagetilor folosim aceeasi metoda isKeyDown:
- Pentru a-l deplasa cu 5 pixeli in sus atunci cand apasam tasta sageata sus.

```
if (Greenfoot.isKeyDown("up"))
    this.setLocation(this.getX(), this.getY()-5);
```

Pentru celelalte directii se programeaza in mod asemanator: down Y + 5, left X – 5, right X + 5

Compilati si executati.

Nu uitati sa folositi tastele pentru directii.

Observatie interesanta !

- Putem programa ursul sa aiba o miscare specifica sau oarecum aleatoare, apoi in continuare sa adaugam si controlul tastaturii.

- In aceasta situatie ursul va fi controlat de taste iar cand nu actionam el va avansa conform programarii initiale.

Interactiunea dintre obiecte

- Acum ursul trebuie sa interactioneze cu albinele.
- Va trebui sa detecteze prezenta unei albine in pozitia in care se afla.
- In metoda Act din Urs continuam:

```
Actor Y = getOneIntersectingObject(Albina.class);
// In Y punem obiectul din clasa Albina cu care se afla in intersectie
 if (Y != null) { // daca a intersectat o Albina, o culege
                  getWorld().removeObject(Y);  //dispare
                  numar_albine++;
                  }
```

- In LumeVasile declaram si initializam contorul:

```
Public static int numar_albine=0;
```

End the Game

- Clasa **Greenfoot** are o metoda Stop pe care o puteti folosi atunci cand vreti sa opriti jocul.

- Puteti dori oprirea jocului:
  - Cand jucatorul a atins un obiectiv propus
  - Cand timpul de joc a fost epuizat
  - Cand instanta atinge o anumita pozitie

In punctul in care doriti terminarea jocului apelati metoda:

```
Greenfoot.stop();
```

Dezvoltari ulterioare

Avem acum o aplicatie functionala care ne permite sa avem control asupra ursului prin intermediul tastaturii si astfel sa avem o interactiune dirijata intre personaje.

- Sunt posibile diverse modificari ulterioare ale scenariului, cum ar fi:
  - ✓ Contorizarea albinelor?
  - ✓ Adaugarea unui stup si vizualizarea numarului de albine in stup?
  - ✓ Adaugarea unui Timer care sa impuna o conditie suplimentara pentru finalizarea jocului (cu succes sau nu)?
  - ✓ Adaugare sunet cand prinde albina sau sunet suav la fiecare albina care inca zboara?
  - ✓ Diversificarea albinelor: albina lucratoare, matca si bondar, avand imagini diferite?
  - ✓ Conditii reale de reusita, cu un anumit numar de albine si o matca si eventual un procent de bondari?

Notiuni prezentate

1. Clase
2. Compilare
3. Instanțe
4. Cod sursă
5. Subclase
6. Superclase

2. GREENFOOT

1. Concepte introductive
2. Mediul integrat de dezvoltare Greenfoot
3. Exemplu de scenariu
4. Referințe bibliografice

Referințe bibliografice

[1] Michael Kölling, Introduction to Programming with Greenfoot, Object-Oriented Programming in Java with Games and Simulations, Pearson Education, August 2009,

http://www.greenfoot.org/book

[2] Greenfoot - An Introduction to OOP, Adrienne Decker, Stephanie Hoeppner, Fran Trees

[3] http://www.greenfoot.org

[4] http://en.wikipedia.org/wiki/Integrated_development_environment

Întrebări?