Programare orientată pe obiecte

#3 JAVA
Greenfoot (partea a II-a)

Adrian Runceanu
www.runceanu.ro/adrian

Curs 3

GREENFOOT.
Utilizarea metodelor, a variabilelor si parametrilor

3. Metode și variabile

1. Depanare
2. Moștenire
3. Variabile de moștenire
4. Metode
5. Apel de metode
6. Parametrii metodelor
7. Tipul returnat
8. Signatura(prototipul) metodelor
9. Variabile

Exemplu de metodă

- Pentru a finaliza o activitate scolara, cum ar fi programele la informatică, există mai multe sub-sarcini:
  1. Studentul Completează tema la informatică
  2. Studentul Merge la școală
  3. Studentul Prezintă temele la profesorul său
- Din cauza experiențelor învățate în școală, combinate cu abilități pre-programate (cum ar fi gândirea), studentul este capabil de a finaliza aceasta activitate.

Exemplu de metodă

- În programare, *fiecare obiect are un set de operații (sau sarcini), pe care le poate efectua*.
- Programatorii scriu un program pentru a spune unui obiect cum și când efectuează activități, cum ar fi:
  - Spune(comandă) unui obiect efectuarea unei acțiuni
  - Întreabă un obiect pentru a afla mai multe informații despre ceea ce face

Metodele sunt o mulțime de operații sau sarcini pe care instanțele unei clase le pot efectua.
Când o metodă este apelată, se va efectua operațiunea sau sarcina specificată în codul sursă.

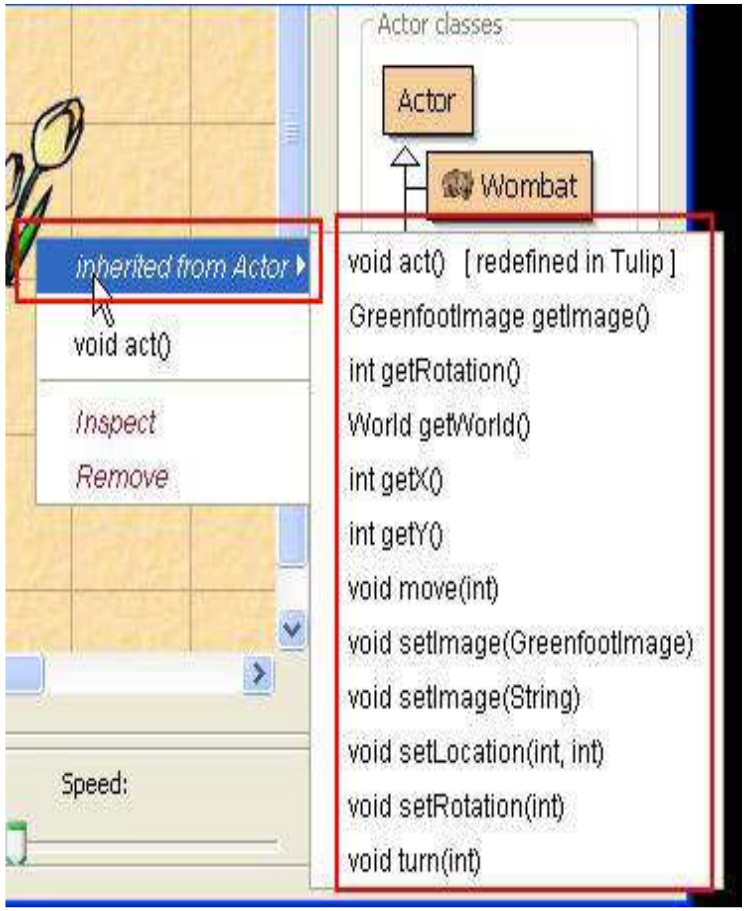
Moștenire

- Obiectele **Greenfoot** moștenesc metodele și proprietățile superclaselor în subclasele lor.
- De exemplu, o instanță Frog va moșteni metodele din superclasa Actor în subclasa Frog.

Moștenirea înseamnă că fiecare subclasă moștenește metodele sale de la superclasa sa.

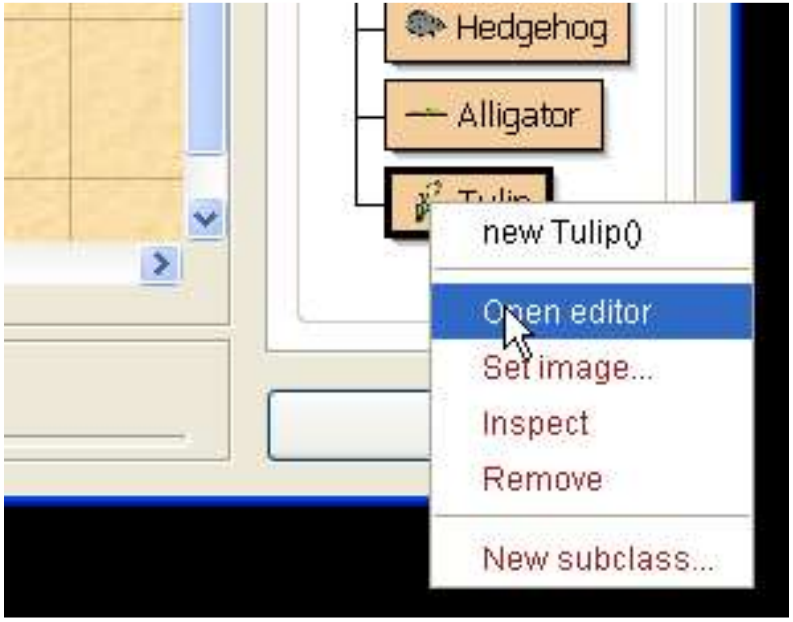
Vizualizarea metodelor in meniul obiectului

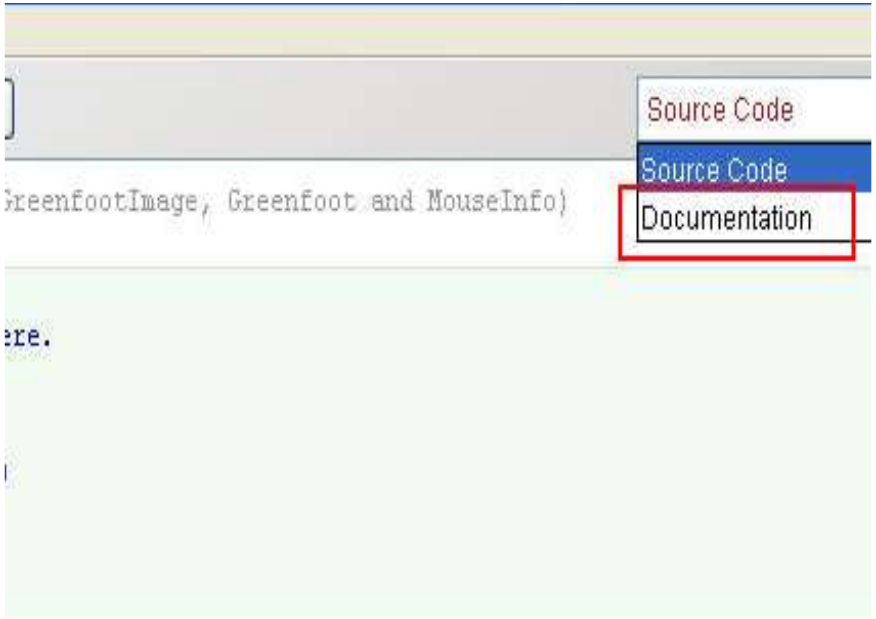
- Meniul obiectului afișează toate metodele de care instanța le moștenește de la superclasa, în subclasa sa.
  - Right click pe instanța obiectului pentru a afișa meniul.
  - Inherited From Actor afișează o listă a metodelor pe care subclasa moștenește de la superclasa Actor.



Pași pentru a vizualiza metodele moștenite in Editorul de cod sursă

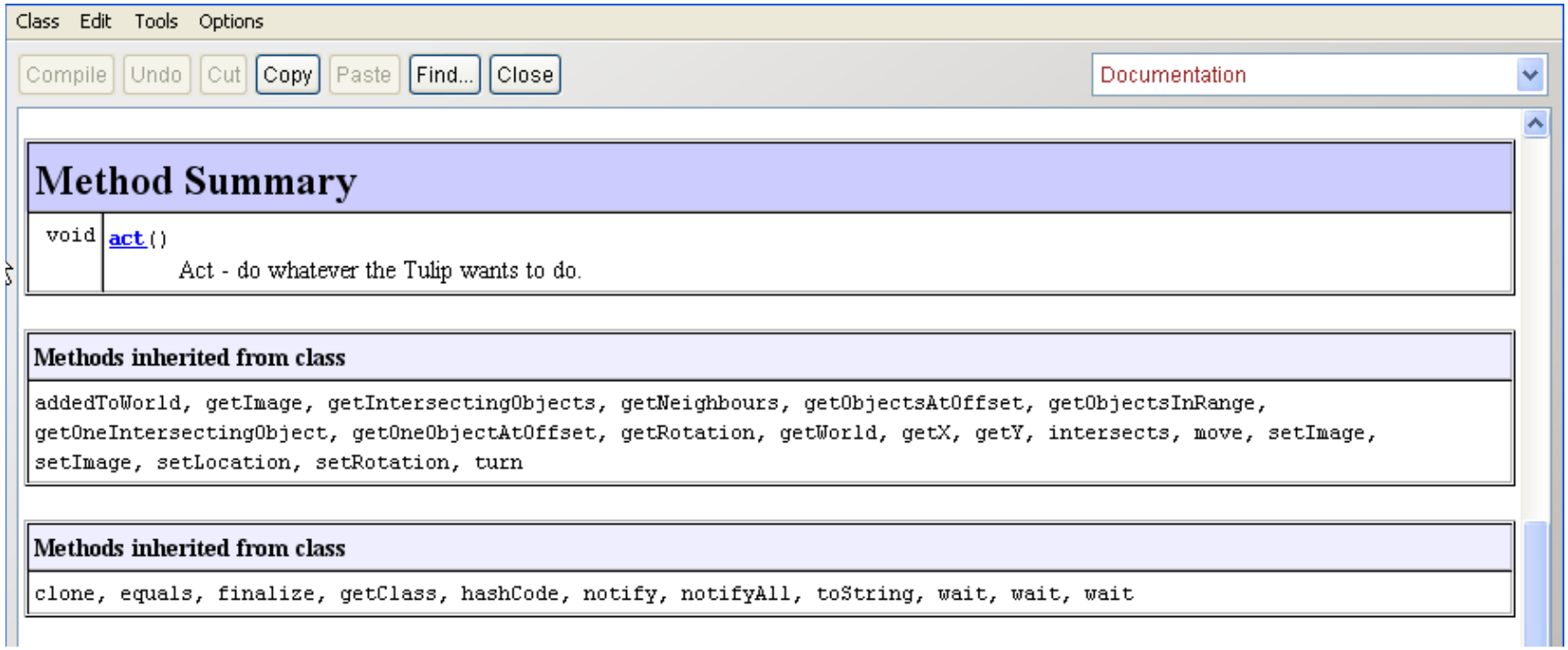
1. Right click pe clasa respectivă.
2. Click Open Editor.
3. În Editorul de cod sursă, selectați meniul drop-down Documentation.
4. Accesează Method Summary.





Rezumatul Metodei

Method Summary (rezumatul metodei) afișează metodele moștenite ale clasei respective



Componentele metodelor

O metodă conține câteva componente care descriu operațiile și activitățile pe care le execută.

- ✓ Tipul returnat: Specifică informația care merge acolo unde se apelează metoda.
- ✓ Numele metodei: Descrie ce anume face metoda respectivă.
- ✓ Lista de parametri: Specifică informațiile care sunt returnate de metodă

Exemple de metode:

```
void move()
void turnLeft()
```

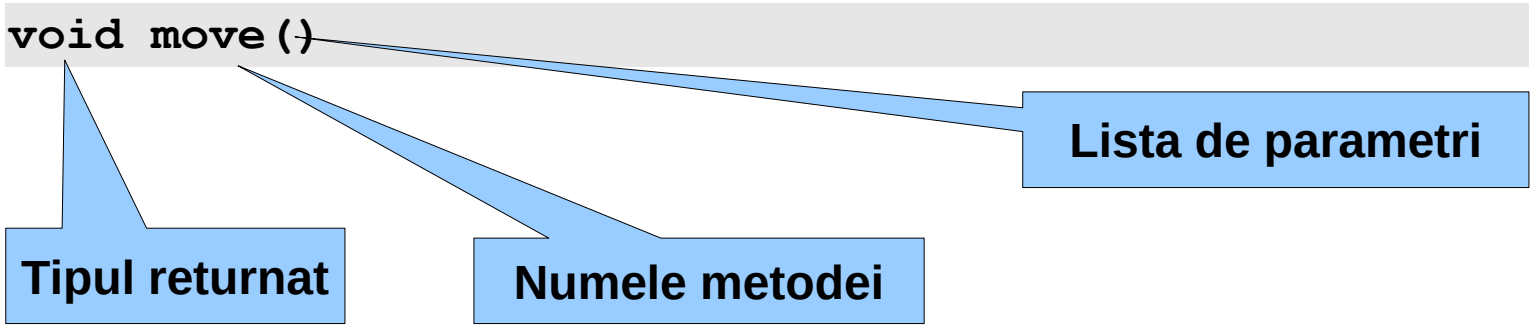
O metodă în Greenfoot apelează comenzile pe care o instanță le execută la o operație sau o activitate. Citește denumirea metodei pentru a putea înțelege ce operație sau activitate se execută la un moment dat.

Prototipul metodelor

Prototipul *unei metode descrie metoda respectivă.*

Acesta conține următoarele componente:

1. Tipul returnat
2. Numele metodei
3. Lista de parametri

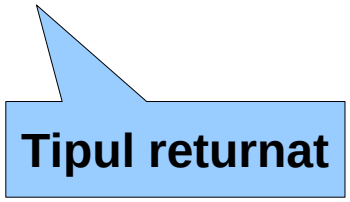


Tipurile returnate de metode

Tipurile returnate de metode sunt:

- Void: Transmite o comandă unui obiect
- Non-void: Pune o întrebare unui obiect

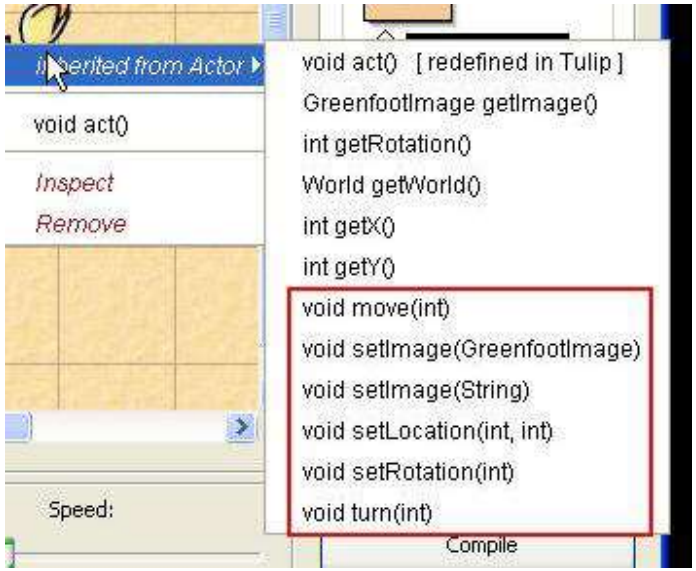
```
void move(int)
```



Metode care returnează tipul void

Metodele cu cuvântul cheie void returnează (transmite) o comandă care execută o acțiune.

- Conține cuvântul cheie "void"
- Nu returnează informații despre obiect
- Se utilizează pentru a defini ceea ce face obiectul



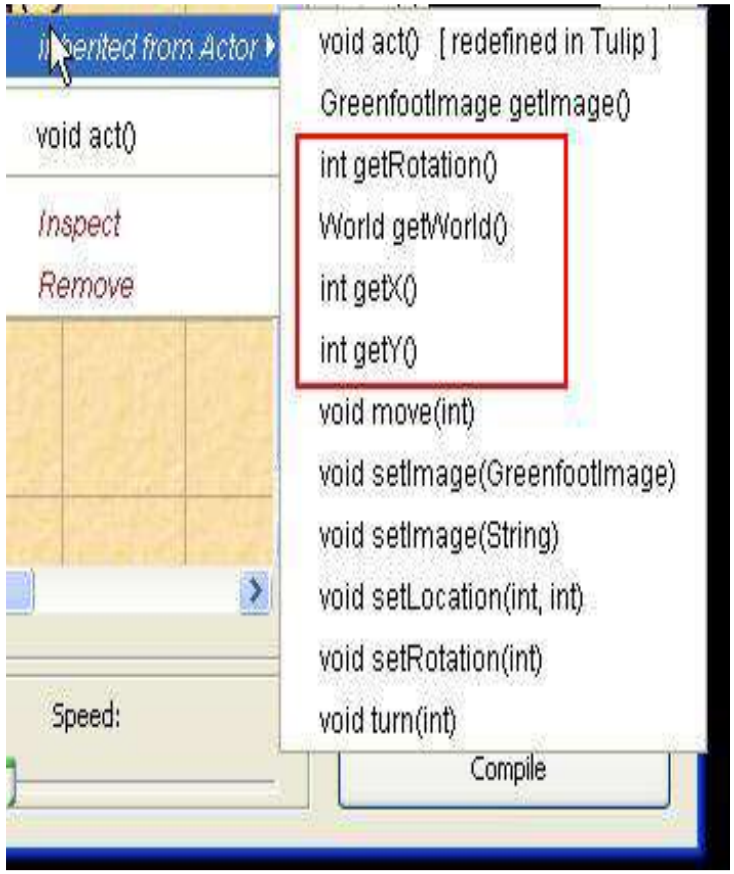
Apelul metodelor care returnează tipul void

Metodele care returnează tipul void se apelează pentru:

- A preciza poziția obiectelor în scenariul tău inițial (punctul de pornire al jocului).
- A comanda ce acțiuni trebuie să efectueze obiectele în joc

Apelul metodelor care returnează tipurile non-void

- Metodele care returneaza tipuri non-void incearca sa afle informatii despre obiect (intreaba obiectul):
  - Prototipul metodei nu include cuvantul cheie “void”
  - Metoda returneaza informatii despre obiect, dar nu actioneaza asupra lui prin modificare sau mutare



Exemple de tipuri non-void

- Integer (afisat ca int)
  - se refera la numere
  - Intreaba obiectul: Cate sunt?

- Boolean
  - Returneaza valoarea true sau false.
  - Tipuri de intrebari la care ar putea raspunde un obiect:
    - Te poti muta?
    - Esti la marginea lumii?

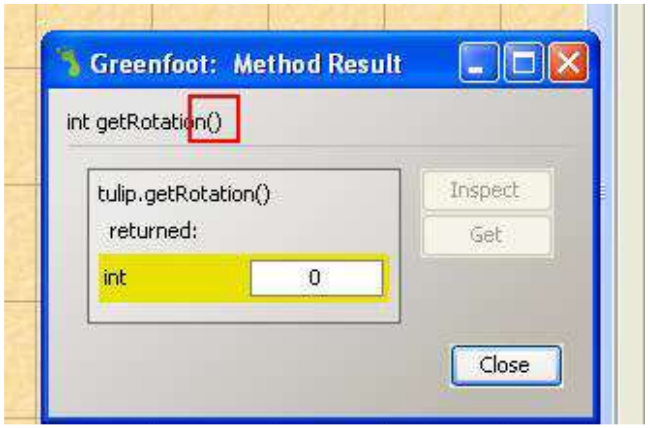
Parametrii metodelor

- Parametrii furnizeaza metodelor informatii suplimentare pentru ca un obiect sa efectueze o activitate, atunci cand informatiile sunt necesare pentru apelul metodelor.
- Parametrii au doua componente:
  - Tipul Parametrului
  - Numele Parametrului

Parametrii se utilizeaza pentru a comanda obiectelor sa se miste, sau pentru a transmite obiectelor ce tip de raspuns este asteptat atunci cand punem o intrebare.

Lista parametrilor metodelor

- Lista parametrilor metodei arata daca metoda necesita informatii suplimentare pentru a fi invocata(apelata) si ce tip de informatii se doreste
  - Lista de parametrii se specifica intre paranteze rotunde.
  - Parametrii pot avea doua stari:
    - Empty: Nu se asteapta date pentru apelul(invocarea) metodei (metoda move)
    - Non-empty: Are date(informatii) si asteapta unul sau mai multi parametri pentru apelul metodei (metoda turn).



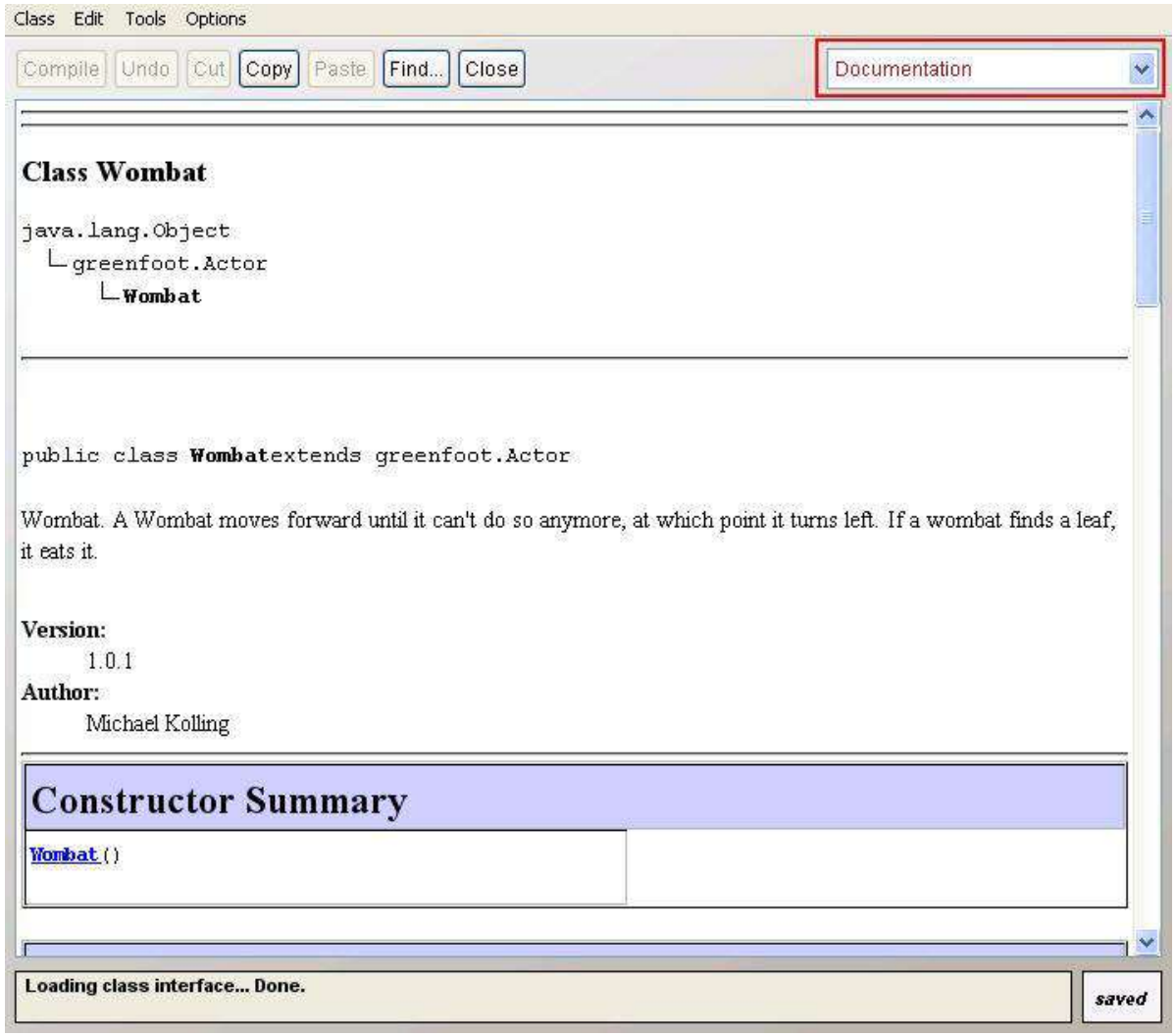
Proprietatile obiectului

- ***Proprietatile obiectului*** *descriu comportamentul instantei si caracteristicile* cum ar fi:
  - Dimensiune
  - Culoare
  - Domeniul(intervalul) de miscari

- Proprietatile pot fi vizualizate si modificate in codul sursa al clasei.

Vizualizarea proprietatilor obiectului

- Vizualizarea proprietatilor unui obiect din documentatia clasei.



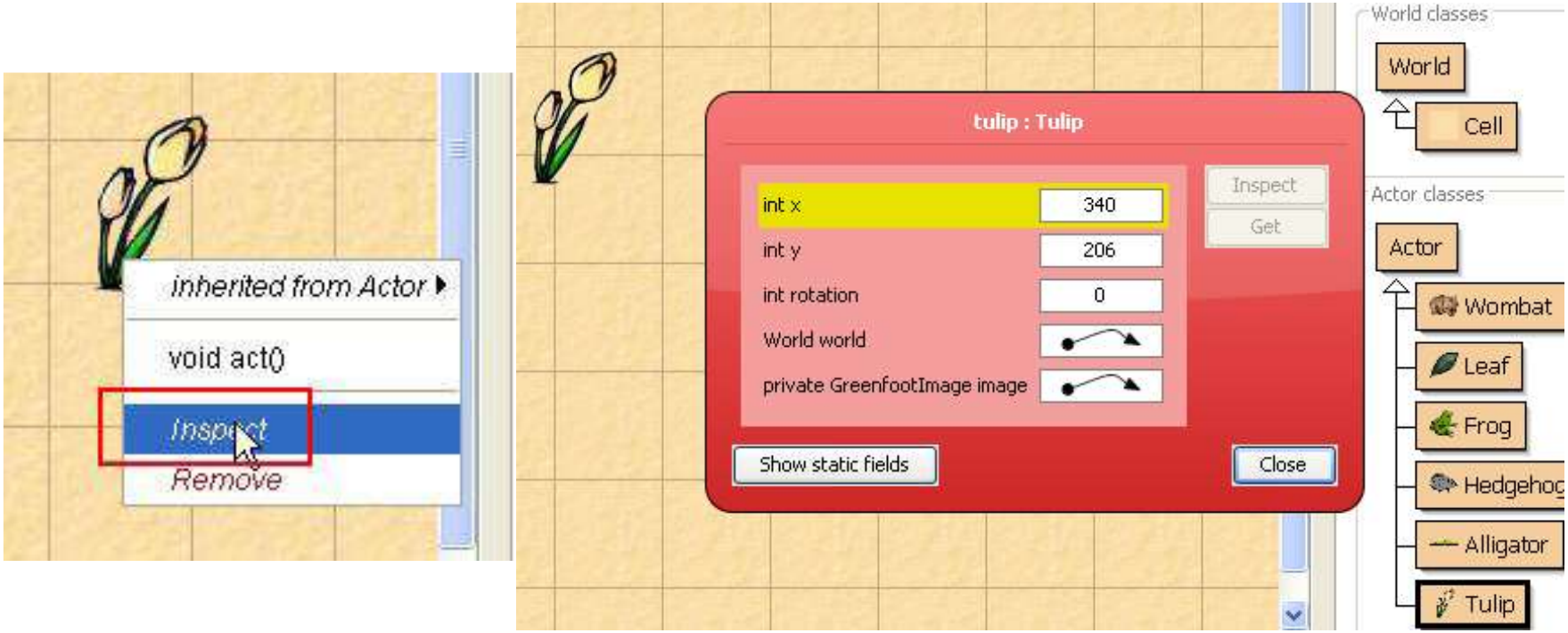
Variabile

- O variabila, sau un camp, permite instantei sa stocheze informatii care vor fi utilizate imediat sau mai tarziu.
- De exemplu, proprietatile obiectului sunt variabile care stocheaza informatii despre instanta, cum ar fi pozitia obiectului in lumea virtuala(world).

Variabilele din instanta se memoreaza in zona de memorie care apartine instantei clasei.
Aceasta memorie poate fi salvata si accesata atata timp cat exista instanta clasei respective.

Vizualizarea variabilelor instantei

- Apasati butonul dreapta pe o instanta, apoi apasati Inspect pentru vizualizarea variabilelor instantei



Sintaxa programului

- **Codul sursa** specifica toate proprietatile si caracteristicile clasei si obiectelor din clasa respectiva.

- Scrieti codul sursa (denumit si sintaxa programului) in editorul de cod sursa pentru a comanda obiectele in metoda Act() din scenariul creat de catre programator.

Afisarea codului sursa al clasei

- Din superclasa World, apasati click dreapta pe clasa si selectati Open Editor pentru afisarea editorului de cod sursa.
- Editorul de cod sursa afiseaza ce executa obiectele din clasa respectiva.

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Duke here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
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()
    {
    {
     // Add your action code here.
    }
}
}
```

Metoda Act()

- In mediul de dezvoltare Greenfoot chiar daca se executa comenzile din metoda Act() sau din butonul Run, obiectele vor face doar ceea ce au fost programate in instructiunile din metoda Act()

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Duke here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
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()
    {
    {
     // Add your action code here.
    }
}
}
```

Structura metodei Act()

- Structura metodei Act() este formata din instructiunile care se scriu intre cele doua acolade.
- In aceasta zona puteti scrie instructiuni pentru instantele clasei care vor fi executate dupa ce se apasa butonul Run

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Duke here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
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()
    {
    {
     // Add your action code here.
    }
}
}
```

Exemplu de metoda Act()

- Apelul metodelor move() si turn() se poate face in metoda Act() pentru ca instantele sa execute miscari de mutare si intoarcere
- Metodele trebuie scrise corect din punct de vedere sintactic, fara caractere lipsa, fara scrieri incorecte de denumiri de instructiuni, deoarece in caz contrar, compilatorul nu va executa acele instructiuni

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Duke here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
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()
    {
    {
     // Add your action code here.
    }
}
}
```

Apelul metodelor in metoda Act()

Pentru apelul metodelor in metoda Act, se scrie urmatoarea secventa de apeluri respectand regulile:

- ✓ Numele metodei cu litere mici
- ✓ Paranteze cu lista de parametri daca este necesara
- ✓ Caracterul **;** la sfarsitul instructiunii

```
public class Bee extends Actor
{
    /**
     * Act - do whatever the Bee wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        move(1);
        turn(15);
    }
}
```

Procesul de depanare a erorilor in Greenfoot

- Scrierea incorecta sau lipsa de caractere in codul sursa va avea ca efect apartia unor mesaje de eroare
- Cand se apasa butonul Compile, compilatorul verifica erorile din codul sursa
- Daca este gasita o eroare, se afiseaza un mesaj de eroare, iar programatorul trebuie sa efectueze modificari in codul sursa inainte ca programul sa fie executat din nou
- Mediul de dezvoltare Greenfoot furnizeaza aceste mesaje de eroare ceea ce face ca, corectarea lor sa fie destul de usoara

Depanarea este procesul de cautare si inlaturare a erorilor dintr-un program

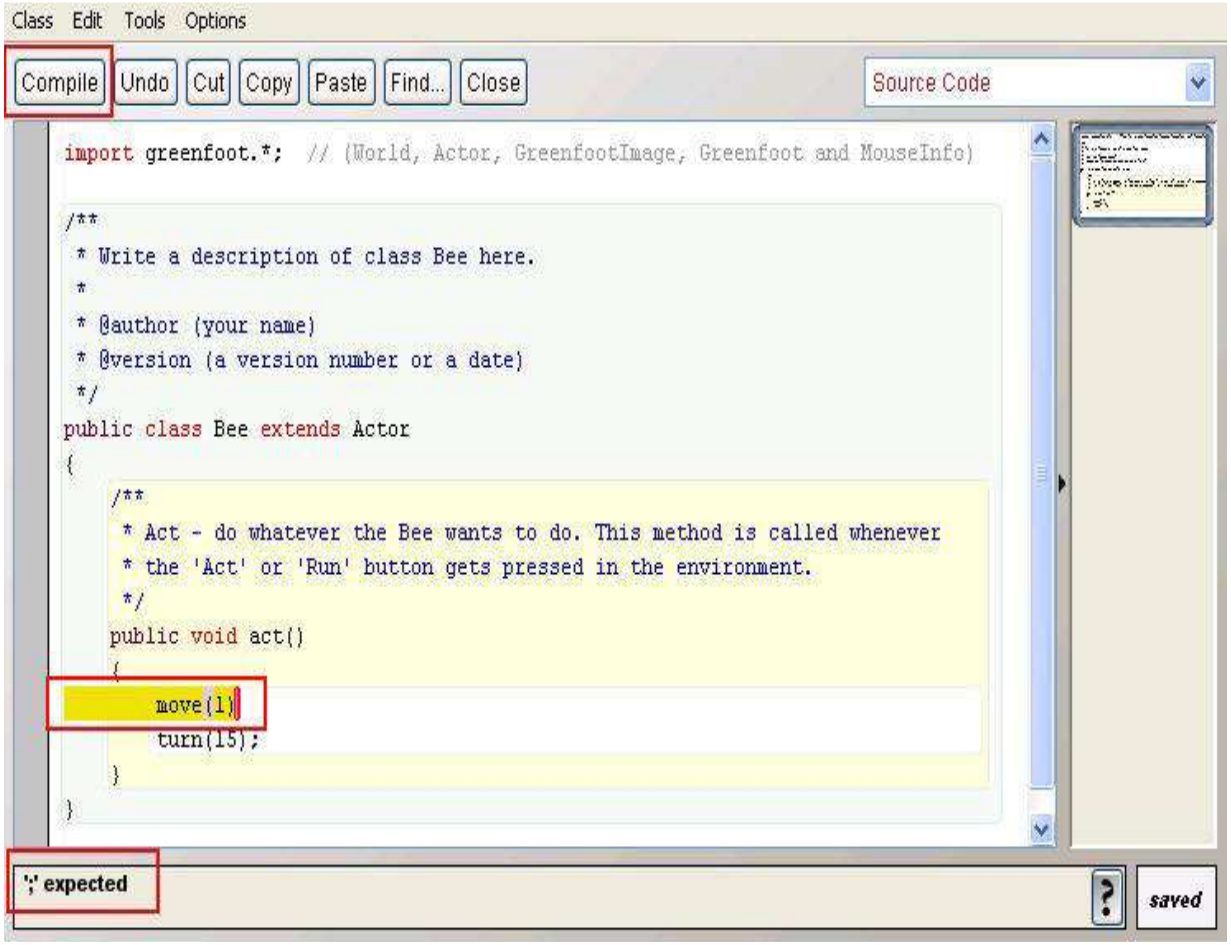
Exemplu de eroare de sintaxa

- La sfarsitul apelului metodei move() lipseste caracterul **;**

```
public class Bee extends Actor
{
    /**
     * Act - do whatever the Bee wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        move(1)
        turn(15);
    }
}
```

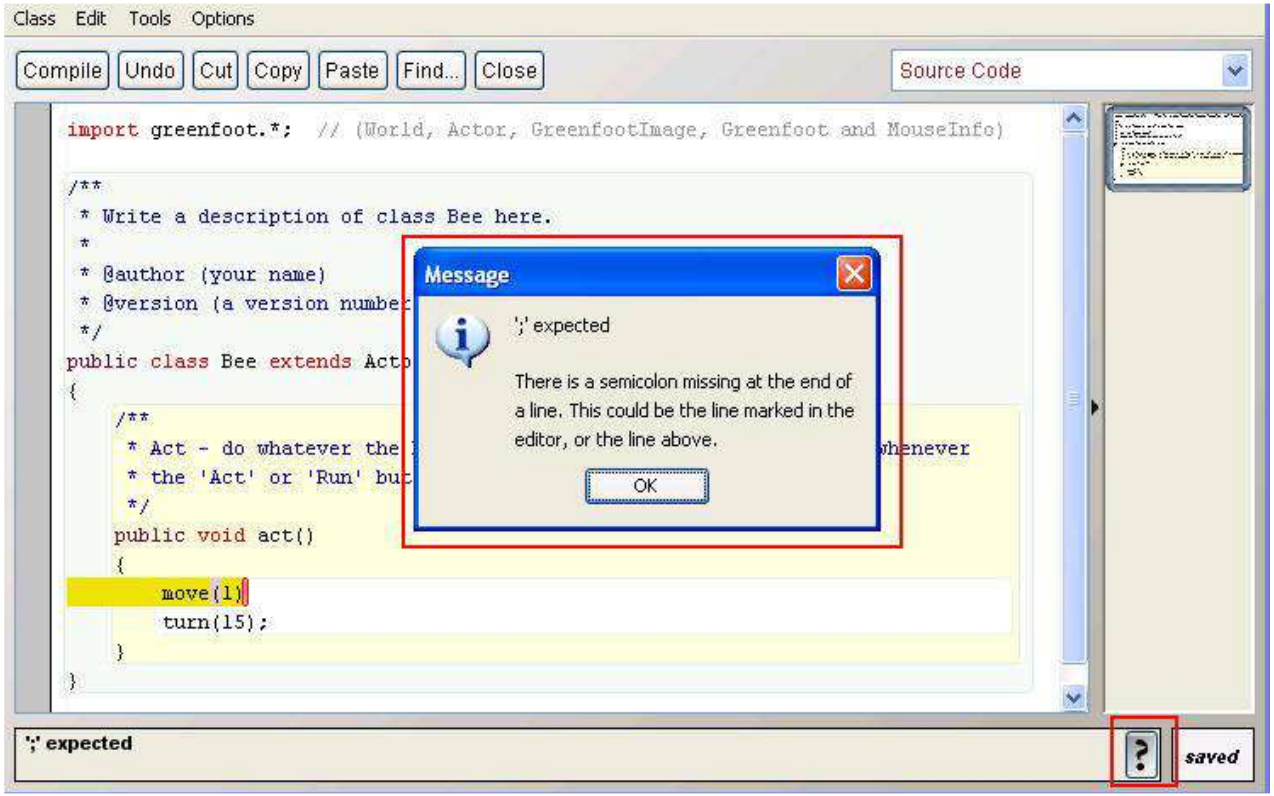
Exemplu de eroare de sintaxa

- Dupa apasarea butonului Compile, apare un mesaj de eroare in zona de jos a ferestrei, iar codul incorect este colorat



Explicatiile eroare de sintaxa

- Apasati pe (**?**) pentru afisarea a mai multor detalii ale mesajului de eroare care apar ca explicatii ale erorii respective
- Nu toate mesajele de eroare sunt usor de inteles!

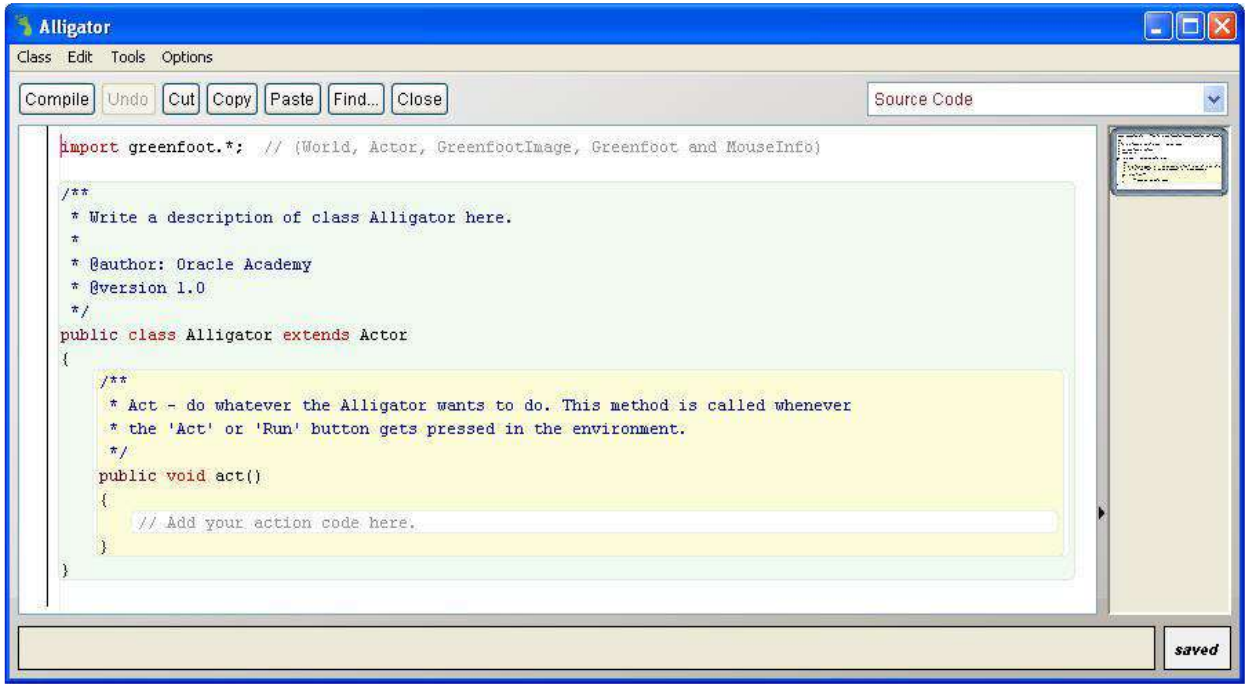


3. Editorul sursa si documentarea aplicatiei

1. Descrierea clasei
2. Comentarii
3. Instructiunea de decizie If
4. Apelul unei metode
5. Analiza orientarii obiectelor
6. Metode secventiale

Editorul de cod sursa

- Codul sursa al aplicatiei scrisa in Greenfoot este gestionat de Editorul de cod sursa.
- Pentru a vizualiza Editorul de cod sursa, apasati click dreapta pe orice clasa din mediul dezvoltare, iar apoi selectati din meniu Open editor



Functiile Editorului de cod sursa

In Editorul de cod sursa se pot efectua urmatoarele:

- Scrie cod sursa pentru programarea instantelor clasei in metoda Act()
- Modifica codul sursa pentru a schimba comportamentul instantelor
- Modifica metodele și proprietățile moștenite de la superclasa
- Modifica metodele create special pentru clasa respectiva de catre programatorul care a scris codul sursa

Componentele Editorului de cod sursa

1	Class Description	Descrierea clasei
2	Act Method	Metoda Act
3	Method Signature	Prototipul metodei
4	Method Body	Corpul metodei
5	Comments	Comentarii
6	Documentation	Documentatia aplicatiei
7	Class Definition	Definitia Clasei

Descrierea clasei

Descrierea clasei este compusa dintr-o multime de comentarii care pot fi modificate pentru a descrie mai bine clasa respectiva.

Aceasta descriere include:

- ✓ O descriere a ce anume face clasa respectiva
- ✓ Numele persoanei care este autorul codului sursa
- ✓ Data ultimei modificari a codului sursa

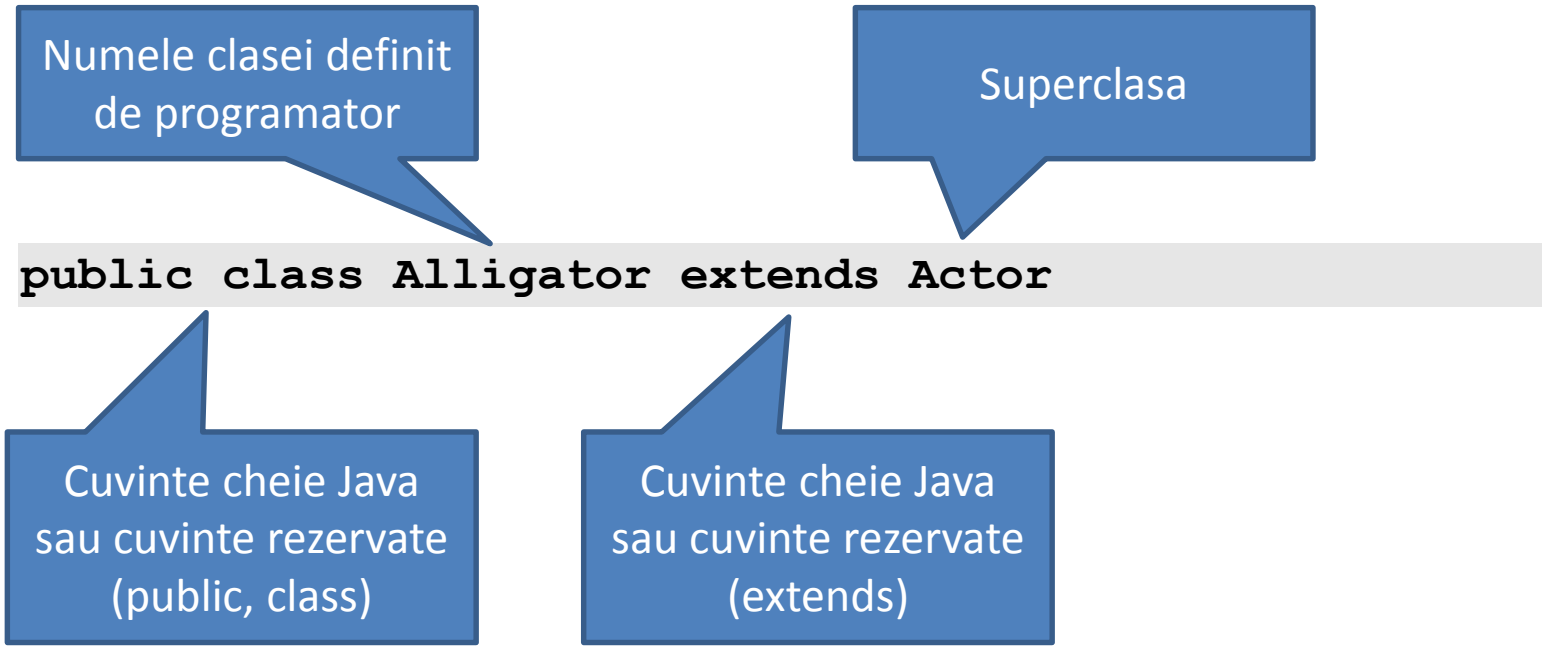
```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Alligator here.
 *
 * @author: Oracle Academy
 * @version 1.0
 */
public class Alligator extends Actor
{
    /**
     * Act - do whatever the Alligator wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }
}
```

Componentele definitiei clasei

Definitia clasei include urmatoarele:

- Cuvinte cheie Java sau cuvinte rezervate
- Numele clasei definit de catre programator
- Numele superclasei din care deriva subclasa respectiva.



Exemplu de definitie a unei clase

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Alligator here.
 *
 * @author: Oracle Academy
 * @version 1.0
 */
public class Alligator extends Actor
{
    /**
     * Act - do whatever the Alligator wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }
}
```

Definitia claselor

Definitia unei clase contine:

1. Definitia variabilelor in care se stocheaza datele permanente ale instantelor clasei respective
2. Definitia constructorilor care initializeaza la inceput o instanta
3. Metode care furnizeaza comportamentele instantelor

Recomandare:

- Folositi un format consistent atunci cand definiti o clasa
  - De exemplu, definiti varibilele mai intai, apoi constructorii, iar la sfarsit metodele clasei

Prototipul metodelor

- Prototipul unei metode descrie ceea ce face metoda respectiva
- Prototipul contine tipul returnat de metoda, numele metodei si lista de parametri

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Alligator here.
 *
 * @author: Oracle Academy
 * @version 1.0
 */
public class Alligator extends Actor
{
    /**
     * Act - do whatever the Alligator wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }
}
```

Comentariile

Comentariile descriu ceea ce se executa in codul sursa:

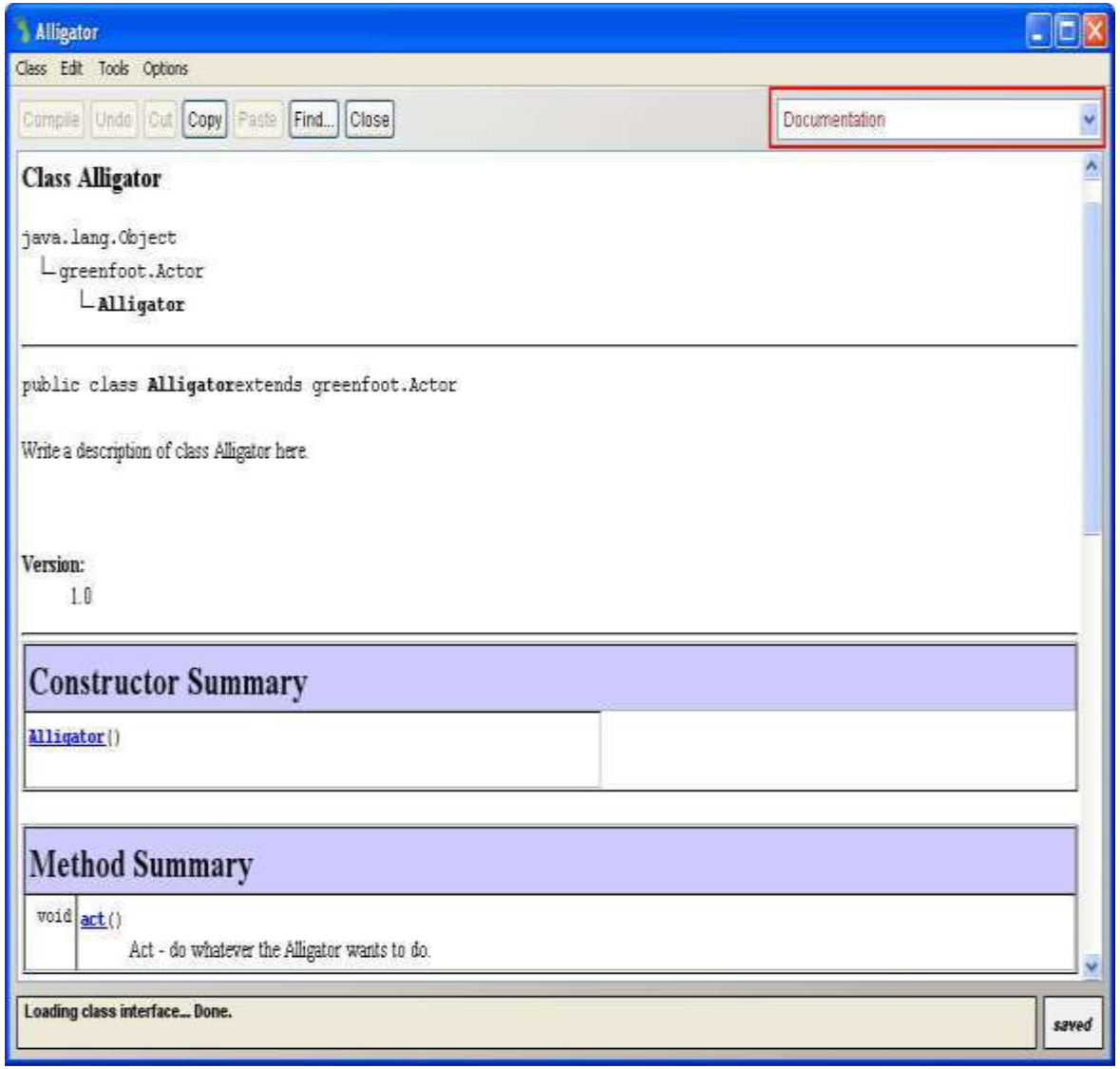
- Nu influenteaza functionarea programului
- Comentariile incep cu combinatia de caractare /**
- Comentariile sunt scrise cu coloarea albastra (numai in Greenfoot)

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Alligator here.
 *
 * @author: Oracle Academy
 * @version 1.0
 */
public class Alligator extends Actor
{
    /**
     * Act - do whatever the Alligator wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }
}
```

Documentatia clasei

- *Documentatia descrie proprietatile clasei*
- Pentru vizualizarea acesteia se selecteaza Documentation din meniul drop-down aflat in partea dreapta-sus a ferestrei editorului de cod sursa



Apelul metodelor in codul sursa

- Metodele trebuie apelate pentru a comanda instantelor anumite actiuni in aplicatia voastra (jocul vostru)
- Apelul metodelor se face prin scrierea lor in metoda Act() in zona identificata de perechea de acolade

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Alligator here.
 *
 * @author: Oracle Academy
 * @version 1.0
 */
public class Alligator extends Actor
{
    /**
     * Act - do whatever the Alligator wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }
}
```

Apelul metodelor in codul sursa

Componentele apelului metodelor sunt:

- Tipul returnat de metoda
  - Variabila care va memora data returnata prin apelul metodei
  - Tipul Void nu necesita o variabila specificata
- Numele metodei
- Lista de parametri pentru a indica tipul de argumente, daca este necesar
- Caracterul ; pentru a marca sfarsitul apelului metodei

Exemplu:

```
public void act()
{
move(10);
turn(50);
}
```

Metode care instruiesc obiecte pentru a efectua anumite actiuni

Numele metodei	Descriere
void move(int distance)	Stabileste numarul de pasi pe care-l poate face obiectul, sau comanda o singura miscare cand se executa metoda Act() sau este apasat butonul Run
void turn(int amount)	Stabileste gradul de intoarcere a obiectului
void act()	Stabileste ce anume executa obiectul ca si actiuni in scenariul aplicatiei. Apelurile metodelor sunt adaugate in aceasta metoda
void setLocation(int x, int y)	Stabileste o noua locatie a unui obiect
void setRotation(int rotation)	Seteaza o noua rotatie pentru obiect

Modalitati de vizualizare a metodelor mostenite de catre orice clasa

1. Vizualizati Documentatia clasei din Greenfoot:
    A. Deschideti Greenfoot
    B. Selectati Help
    C. Selectati Documentatia clasei

2. Vizualizati Documentatia din biblioteca Java:
    A. Deschideti Greenfoot
    B. Selectati Help
    C. Selectati Documentatia din biblioteca Java

Metode secventiale

- Metodele secventiale sunt toate metodele care se executa in mediul de dezvoltare Greenfoot in ordinea in care au fost scrise in program
- Aceste metode fac posibil ca un obiect sa execute activitati in mod secvential, activitati cum ar fi: alergare, saritura, joc sau aparitia unui sunet atunci cand ceva explodeaza
- Obiectele pot fi programate pentru a executa secvential metodele de ori cate ori ori se apasa butonul Act.

Relatii Dacă-Atunci (If-Then)

- Multe lucruri din jurul nostru au o relație cauză-efect de tipul “dacă-atunci”:
  - Dacă telefonul sună, atunci trebuie să răspundem. Dacă nu sună, atunci nu are rost să răspundem
  - Dacă o floare începe sp se usuce atunci o udăm. Dacă o floare este sănătoasă atunci nu o udăm

Instructiunea de decizie If

- O instructiune de decizie if este folosita pentru a transmite programului sa execute o multime de instructiuni doar atunci si numai atunci cand o anumita conditie este adevarata

```
if (condition)
{
        instruction;
        instruction;
        …
}
```

Componentele instructiunii If

- Instructiunea if contine o conditie, care este o expresie care va fi evaluata la valoarea de adevarata sau falsa si unul sau mai multe apeluri de metode care se executa daca conditia este adevarata.

```
public class Alligator extends Actor
{
    /**
     * Act - do whatever the Alligator wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        move(1);
        if (Greenfoot.isKeyDown("left"))
        {
            turn(-2);
        }
        if (Greenfoot.isKeyDown("right"))
        {
            turn(2);
        }
    }
}
```

Exemplu de instructiune If

In urmatorul exemplu avem:

- Tastele left si right de pe tastatura prin intermediul carora obiectul de deplaseaza spre stanga sau spre dreapta
- Daca conditia este falsa, apelurile metodelor scrise in instructiunea if nu se executa
- Metoda de intoarcere (turn) este executata in functie de evaluarea conditiei din instructiunea if

```
public void act()
{
    move(1);
    if (Greenfoot.isKeyDown("left"))
    {
        turn(-2);
    }
    if (Greenfoot.isKeyDown("right"))
    {
        turn(2);
    }
}
```

Metoda isKeyDown

- Metoda isKeyDown este o metoda predefinita in mediul de dezvoltare Greenfoot care “asculta” si determina care tasta a fost apasata in timpul executiei unui program
- Apelul acestei metode se face prin intermediul notatiei “.”

Atunci cand o metoda nu se afla in clasa sau in clasa mostenita pe care o utilizam in programare, atunci se specifica clasa sau obiectul care dupa care se pune caracterul “.” si apoi apelul metodei respective. Aceasta tehnica de programare se numeste notatia “.”.

Orientarea obiectului in lumea reala

- Asa cum noi ne miscam in lumea in care traim, este important pentru noi sa cunoastem orientarea noastra, sensul sau directia de miscare
  - Cand conducem o masina, trebuie întotdeauna sa stim este pe partea corecta a drumului!
  - Cand un avion zboara in aer, acesta trebuie sa stie care este localizarea sa fata de alte avioane, pentru a putea evita o coliziune
  - Cand introduci o localizare pe o harta in telefonul mobil, receptionezi coordonatele care iti spun unde te afli si adresa ta

Afisarea orientarii obiectului

- Metodele ne pot spune cum un obiect este pozitionat in lumea virtuala (World), relativ la ele insusi si relativ la alte obiecte

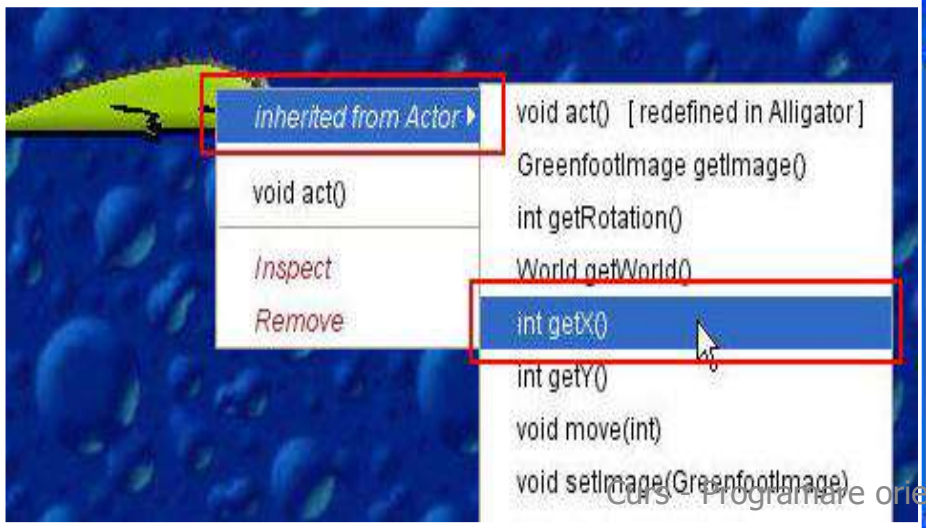
- Puteti apela o metoda:
  - Pentru a afla (intreba) obiectul despre orientarea lui, folosind un tip de date, de exemplu tipul boolean
  - In mediul de dezvoltare pentru a afla cum este orientat obiectul in cadrul scenariului

Metode care returneaza informatii despre orientarea obiectelor

Numele metodei	Descriere
int getRotation()	Returneaza rotatia curenta a obiectului
World getWorld()	Returneaza numele lumii virtuale in care de afla obiectul
int getX()	Returneaza coordonata x a locatiei curente a obiectului
int getY()	Returneaza coordonata y a locatiei curente a obiectului

Pasi care trebuie urmati pentru apelul unei metode care afiseaza orientarea obiectului

1. Right click pe instant lumii virtuale (World)
2. Selecteaza Inherited from Actor pentru a vizualiza metodele sale
3. Selecteaza o metoda cu un anumit tip de date pentru a afisa informatii despre orientarea obiectului
4. Rezultatul cu locatia obiectului va fi afisat. Retine rezultatul returat si apoi apasa Close.



Întrebări?