Programare orientată pe obiecte

#6 JAVA
Greenfoot (partea a V-a)

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

Curs 6

GREENFOOT.
Instructiuni repetitive.
Variabile. Siruri de caractere

Instructiuni. Variabile. Siruri de caractere

1. Crearea unei bucle *while* într-un constructor pentru a construi o lume virtuala
2. Descrierea functionarii unei bucle infinite
3. Crearea unei expresii prin intermediul operatorilor logici
4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala
5. Folosirea variabilelor de tip șir de caractere pentru a stoca și concatena şiruri de caractere

Greenfoot

Instructiuni repetitive (LOOPS)

➢ **Instructiunile repetitive** (numite si **bucle**) sunt cel mai eficient mod de a crea mai multe instanţe ale unei clase.

➢ O buclă (instructiune repetitiva – LOOP) este o instrucţiune care poate executa o secțiune de cod de mai multe ori.
➢ Există mai multe tipuri de bucle în limbajul Java.

1. Instructiuni repetitive – instructiunea ***while***

- Instructiunea repetitiva ***while*** execută o instrucţiune sau o multime de instrucțiuni de mai multe ori.

De exemplu, cu o buclă WHILE, puteți:

- Să creați 50 de instanţe dintr-o dată
- Să executaţi o metodă de 10.000 de ori

1. Instructiuni repetitive – instructiunea ***while***

Structura unei instructiuni ***while*** este compusa din:

- Cuvântul cheie JAVA, ***while***
- Condiţia logica scrisa între paranteze rotunde
- Una sau mai mai multe instrucţiuni

```
while (condiție logica)
{
        instrucţiune;
}
```

1. Instructiuni repetitive – instructiunea ***while***

Execuţia instructiunii repetitive ***while***:

Pentru a putea executa o instructiune repetitiva ***while*** avem nevoie de urmatoarele elemente:

- Variabilă Loop: Un contor care spune de câte ori se execută instructiunea repetitiva (adesea numit i)
- Operatorii de control
- Variabilă locală

1. Instructiuni repetitive – instructiunea ***while***

Variabile locale

- De obicei se foloseste cel putin o variabila locala într-o instructiune repetitiva
- Deși este similară cu o variablă globală, este diferită de aceasta pentru că:
  - Este declarată în interiorul metodei, nu la începutul clasei
  - Nu are un modificator de vizibilitate (*public* sau *privat*), în fața definiției sale.
  - Există numai până când metoda termină de executat, apoi fiind ștearsă din memorie.

O **variabilă locală** este o variabilă declarată în interiorul unei metode pentru stocarea temporară a valorilor, cum ar fi referințe la obiecte sau numere întregi.

1. Instructiuni repetitive – instructiunea ***while***

Declararea unei variabile locale

- Pentru a crea o instructiune repetitive ***while***, în primul rând declarăm o variabilă locală și îi atribuim o valoare.
- Pentru a declara o variabilă locală:
  - Declarăm tipul variabilei (număr întreg sau obiect de referință)
  - Denumim variabila
  - Iniţializăm variabila cu un număr (de obicei zero)

Exemplu:

```
int i = 0;
```

1. Instructiuni repetitive – instructiunea ***while***

Stabilirea condiţiei logice:

- Dupa inițializarea variabilei, se creează condiția logica care specifică de câte ori ar trebui să fie executat corpul instructiunii.
- Se foloseste un operator de control pentru a opri execuția atunci când ajunge la numărul de execuții specificate.
- Exemplu:
  - Execută corpul instructiunii cat timp numărul de execuții este mai mic strict decât 10.
  - Când instructiunea a fost executată de 10 ori (de la 0 la 9), se oprește.

```
int i = 0;
while (i < 10){
}
```

1. Instructiuni repetitive – instructiunea ***while***

Adaugarea instrucţiunilor de execuţie:

Între paranteze (acolade), se introduc instrucţiunile de executat.

Exemplu:

- Adaugă 10 obiecte Duke cu o tastă specifică și un fișier de sunet atașat la fiecare obiect.

```
int i = 0;
while (i < 10) {
        addObject (new Duke (“k”, “test.wav”),150, 100);
}
```

1. Instructiuni repetitive – instructiunea ***while***

Incrementarea variabilei contor:

Se incrementează variabila contor, astfel:

- Inserează o instrucţiune la sfârșitul corpului instructiunii pentru a crește variabila contor cu 1 de fiecare dată când instructiunea este executată.
- Utilizează acolada închisa pentru a încheia instrucţiunea.

Acest lucru va schimba variabila cu fiecare repetare, în aşa fel încât instructiunea să nu se repete la nesfarsit (*bucla infinita*).

```
int i = 0;
while (i < 10)
{
        addObject (new Duke (“k”, “test.wav”), 150, 150);
        i = i + 1;
}
```

1. Instructiuni repetitive – instructiunea ***while***

Exemplu de instructiune repetitiva ***while***:
Această instructiune repetitiva ***while*** a fost introdusă în constructorul WORLD și creează 10 obiecte de tip Duke când se inițializeaza obiectul WORLD

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

/**
 * Write a description of class DukeWorld here.
 *
 * @author Oracle Academy
 * @version JF_S03_L05
 */
public class DukeWorld extends World
{
    /**
     * This constructor creates a new world and the objects that start the
     * game.
     */
    public DukeWorld()
    {
        super(600, 400, 1);
        int i = 0;
        while (i < 10)
        {
            addObject (new Duke("k", "test.wav"), 150, 100);
            i = i + 1;
        }
    }
}
```

1. Instructiuni repetitive – instructiunea ***while***

Adaugarea obiectelor folosind instructiunea ***while***:

- În exemplul anterior, când constructorul este executat, toate instanțele sunt plasate în aceleași coordonate.
- Acest lucru le face să se așeze pe unele peste altele.
- Creați o expresie care calculează dimensiunea unei instanțe, iar apoi pune instanțe ulterioare la coordonate diferite.

1. Instructiuni repetitive – instructiunea ***while***

Se calculează adaugarea de instanțe:

- Pentru a programa instanțele să aterizeze pe coordonate diferite, înlocuiți coordonata x fixată pentru lățimea obiectului cu o expresie care include:
  - Variabila i.
  - Operatorul de multiplicare (*).
  - Număr întreg fixat astfel încât primul obiect aterizează in zona ecranului.

```
public class DukeWorld extends World
{
    /**
     * This constructor creates a new world and the objects that start the
     * game.
     */
    public DukeWorld()
    {
        super(600, 400, 1);
        int i = 0;
        while (i < 10)
        {
            addObject (new Duke("k", "test.wav"), i*150 + 30, 100);
            i = i + 1;
        }
    }
}
```

Instructiuni. Variabile. Siruri de caractere

1. Crearea unei bucle *while* într-un constructor pentru a construi o lume virtuala
2. Descrierea functionarii unei bucle infinite
3. Crearea unei expresii prin intermediul operatorilor logici
4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala
5. Folosirea variabilelor de tip șir de caractere pentru a stoca și concatena şiruri de caractere

Greenfoot

2. Descrierea functionarii unei bucle infinite

Bucle Infinite:

- În cazul în care componenta a unei buclei nu este stabilita corect, atunci bucla continuă sa se execute și nu se oprește niciodată.
- Buclele infinite sunt o problemă comună în programare.
- O buclă infinită apare dacă:
  - Variabila nu se schimbă niciodată.
  - Condiția logica rămâne întotdeauna adevărată.

O buclă infinită are loc atunci când bucla continuă să se execute și nu se oprește pentru că finalul acesteia nu este stabilit.

Instructiuni. Variabile. Siruri de caractere

1. Crearea unei bucle *while* într-un constructor pentru a construi o lume virtuala
2. Descrierea functionarii unei bucle infinite
3. Crearea unei expresii prin intermediul operatorilor logici
4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala
5. Folosirea variabilelor de tip șir de caractere pentru a stoca și concatena şiruri de caractere

3. Crearea unei expresii prin intermediul operatorilor logici

Animarea obiectelor de la tastatură:

- Un alt mod de a anima un obiect este de a face ca obiectul respectiv să-si schimbe imaginea pe care acesta o afişează atunci când este apăsată o tastă

- Algoritmul pentru această acțiune:
  - Interschimbare între două imagini atunci când este apăsată o tastă.
  - Când tasta este apăsată, afişează imagine1.
  - Când nu este apăsată tasta, afişează imagine2.
  - Obiectul trebuie să-și amintească dacă o anumita tasta este apăsată sau nu.
  - În caz contrar, obiectul care se afişează va fi schimbat rapid, iar tasta nu va fi capabilă să-l controleze.

3. Crearea unei expresii prin intermediul operatorilor logici

Exemplu în vederea controlului de la tastatura:

- Obiectul Duke ar trebui să facă cu mâna dacă o tastă este apăsată.
- Două imagini sunt salvate în scenariu:
  - Una cu brațul lui Duke în sus
  - şi una cu brațul în jos

Algoritmul pentru această acțiune:

- Dacă brațul lui Duke este sus, iar tasta este apăsată, atunci arată imaginea cu brațul lui Duke jos.
- Ţine minte că braţul lui Duke este jos la început.

3. Crearea unei expresii prin intermediul operatorilor logici

Precizați imaginea de afișat:

- În primul rând, scrie codul în metoda act() a clasei pentru a specifica imaginea de afişat dacă tasta este apăsată, sau nu.
- Utilizați următoarele metode:
  - metoda **isKeyDown** (utilizează notaţie **dot**)
  - metoda **setImage**

```
public void act()
{
        if (Greenfoot.isKeyDown("k")){
                setImage ("Duke.PNG");
        }
        else {
                setImage ("DukeDown.PNG");
        }
}
```

3. Crearea unei expresii prin intermediul operatorilor logici

Declararea unei variabile cu denumirea isDown:

Se declară variabila isDown în codul sursă al clasei și se atribuie lui Duke, pentru a-l face pe Duke să-și amintească dacă tasta este apăsată sau nu.

Verifica condiţia adevărat/fals:

- True - atunci când tasta este apăsată.
- False - atunci când tasta nu este apăsată.

3. Crearea unei expresii prin intermediul operatorilor logici

Exemplu de aplicare a variabilei isDown:

Variabila isDown este declarată și setată cu valoarea false, pentru că scenariul începe cu tasta neapăsată.

```
public class Duke extends Animal
{
    private GreenfootImage image1;
    private GreenfootImage image2;
    private boolean isDown;
    private String key;
    private String sound;
    /**
     * Create a Duke and initialize his two images. Link Duke to a specific keyboard
     * key and sound.
     */
    public Duke(String keyName, String soundFile)
    {
        key = keyName;
        sound = soundFile;
        image1 = new GreenfootImage("Duke.PNG");
        image2 = new GreenfootImage("DukeDown.PNG");
        setImage(image1);
        isDown = false;
    }
```

3. Crearea unei expresii prin intermediul operatorilor logici

Operatorii Logici

- Pentru a testa dacă brațul lui Duke este sus sau jos atunci când este apăsată o tastă, acest lucru necesită utilizarea:
  - Unor expresii booleene multiple care să exprime dacă una sau ambele sunt adevărate sau false.
  - Operatori logici pentru a conecta expresiile booleene.

- De exemplu, prima instrucţiune:

"Dacă brațul lui Duke nu este în jos, și tasta “**d**” este apăsata..." s-ar coda ca:

```
if (!isDown && Greenfoot.isKeyDown(“d”) )
```

3. Crearea unei expresii prin intermediul operatorilor logici

Tipuri de Operatori Logici

Operatorii logici pot fi folosiţi pentru a combina mai multe expresii booleene într-o singură expresie boolean.

Operator Logic	Ce înseamnă	Definiție
Semnul exclamarii (!)	**NOT**	*Inversează valoarea unei expresii booleene* (în cazul în care b este adevărat, !b este fals. Dacă b este fals, !b este adevărat).
Ampersand dublu (&&)	**AND**	Combină două valori booleene, și returnează o valoare booleană care este *adevărată dacă și numai dacă ambii săi operanzi sunt adevăraţi*.
Semnul linii verticale (||)	**OR**	Combină două variabile sau expresii booleene și returnează un rezultat care este *adevărată dacă unul sau ambii operanzi sunt adevăraţi*.

3. Crearea unei expresii prin intermediul operatorilor logici

Exemplu de operatori logici:

Operatorii logici care setează imaginea care apare în cazul în care tasta "**d**" este apăsată sau nu.

```
/**
 * 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(2);
    if (!isDown && Greenfoot.isKeyDown("d"))
    {
        setImage ("Duke.PNG");
        isDown = true;
    }
    if (isDown && !Greenfoot.isKeyDown("d"))
    {
        setImage ("DukeDown.PNG");
        isDown = false;
    }
    lookForCode();
}
```

3. Crearea unei expresii prin intermediul operatorilor logici

Adaugarea sunetului in program:

- Odată ce instrucţiunea este programată pentru a-l anima pe Duke, ultimul pas este de a programa o instrucţiune care îl determină pe Duke să facă un sunet atunci când este apăsată tasta "d", în plus față de mișcare brațului său.

- Definiți metoda de a reda sunetul, astfel încât să o puteți apela în metoda de act() atunci când tasta respectivă este apăsată.

3. Crearea unei expresii prin intermediul operatorilor logici

Definiți metoda de redare sunetului:

- În primul rând, să se definească o metodă în clasa numita play().
- Scrie metoda sub metoda act(), așa cum poţi vedea mai jos.

Această metodă:

- Apelează metoda playSound din clasa Greenfoot folosind notația punct în corpul instrucţiunii ***if***
- Include numele fișierului sunet ce urmează a fi redat.

```
/**
 * Play a sound.
 */
public void play()
{
    if (Greenfoot.isKeyDown("d"))
    {
        Greenfoot.playSound("test.wav");
    }
}
```

3. Crearea unei expresii prin intermediul operatorilor logici

Introduceți metoda de redare a sunetului în metoda act():

- Introduceţi apelul metodei play() în una dintre instrucţiunile **if** astfel încât sa fie redată atunci când este apăsată o tastă.
- Introduceţi apelul metodei play() sub instrucţiunea **if** astfel încât să se redea în continuu în timpul jocului.

```
/**
 * 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(2);
    if (!isDown && Greenfoot.isKeyDown("d"))
    {
        setImage ("Duke.PNG");
        play();
        isDown = true;
    }
    if (isDown && !Greenfoot.isKeyDown("d"))
    {
        setImage ("DukeDown.PNG");
        isDown = false;
    }
    lookForCode();
}
```

Instructiuni. Variabile. Siruri de caractere

1. Crearea unei bucle *while* într-un constructor pentru a construi o lume virtuala
2. Descrierea functionarii unei bucle infinite
3. Crearea unei expresii prin intermediul operatorilor logici
4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala
5. Folosirea variabilelor de tip șir de caractere pentru a stoca și concatena şiruri de caractere

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Tablouri unidimensionale (Vectori)

- Atunci când creați mai multe instanțe folosind un constructor de tip buclă ***while***, fiecare primește același fișier de sunet și aceeaşi atribuire de tastă.
- In cele mai multe situații, acest lucru nu este ideal. Instanţele pot avea nevoie să reacționeze la diferite taste, sau să redea diferite sunete.
- Folosind un vector, aveți posibilitatea să rețineți si să accesaţi mai multe variabile și să atribuiţi diferite valori unor noi instanţe de fiecare dată când sunt create.

Un **tablou unidimensional (vector)** este un obiect care reține mai multe variabile. Un index poate fi folosit pentru a accesa variabilele.

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Cum se reţin valori in variabile:

O simplă variabilă de tip şir de caractere numită "keyNames" este un container care reţine o valoare si anume numele unei singure taste.

```
String keyNames;
```

Continutul variabilei este tasta “a”:

```
a
```

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Cum se reţin valori intr-o varibila de tip vector (tablou unidimensional):

Un obiect de tip vector poate reține mai multe variabile. Acest vector denumit *keyNames* poate deține mai multe variabile.

String []								
0	1	2	3	4	5	6	7	8
“g”	“r”	“e”	“e”	“n”	“f”	“o”	“o”	“t”

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Declararea unei variabile de tip vector:

Declararea unui obiect de tip vector, contine urmatoarele elemente:

- Tipul elementului:
  - String [] pentru un vector de siruri de caractere.
  - int [] pentru un vector de numere întregi.
- Paranteze drepte [] pentru a indica faptul că această variabilă este un vector
- Atribuirea unor valori variabilei respective
- Expresie care creează obiectul de tip vector și îi adauga un număr nelimitat de siruri de caractere sau numere întregi.

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Exemplu de variabila de tip vector:

- Variabila de tip șir de caractere keyNames este creată.
- Șirurile “g”, “r”, “e”, “e”, “n”, “f”, “o”, “o” și “t” sunt atribuite in variabila respetiva.
- Un obiect de tip vector este atribuit variabilei keyNames.

```
String [] keyNames;
keyNames = {“g”, “r”, “e”, “e”, “n”, “f”, “o”, “o”, “t”};
```

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Accesarea elementelor dintr-un vector:

Utilizați un index pentru a accesa elementele din obiectul vector.

Pentru a utiliza un index:

-Fiecare element are un indice, începând de la poziția zero [0].

-Poziția fiecărui element crește cu 1.

Elementele sunt accesate folosind paranteze drepte [] și un index pentru a specifica ce element al vectorului trebuie accesat.
Un index este un număr de poziție în obiectul de tip vector.

String []								
0	1	2	3	4	5	6	7	8
“g”	“r”	“e”	“e”	“n”	“f”	“o”	“o”	“t”

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Accesarea elementelor dintr-un vector:

Pentru a accesa un element dintr-o variabila de tip vector, atașați indicele pentru acel element în paranteze drepte la numele variabilei.
Instrucțiunea keyNames[3] accesează elementul aflat la indexul 3 - șirul “e”. Acesta este al patrulea element din vector.

String []								
0	1	2	3	4	5	6	7	8
“g”	“r”	“e”	“e”	“n”	“f”	“o”	“o”	“t”

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Redarea diferitelor sunete când anumite taste sunt apăsate:

- Pentru a face o instanță capabilă de a reda o varietate de sunete bazate pe apăsarea unei respective taste, se pot creea doi vectori în clasa World.
- Vectori contin:
  - Numele tastelor pentru instanțele lui Duke.
  - Numele fișierelor de sunet pentru instanțele lui Duke.

- Declară câmpuri in clasa World pentru vectori
- Memoreaza vectorii cu valorile adaugate

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Crearea vectorilor:

Creați doi vectori care să fie în clasa World, indicând denumirile tastelor care se apasa și fișierele de tip sunet utilizate.

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

/**
 * Write a description of class DukeWorld here.
 *
 * @author Oracle Academy
 * @version JF_S03_L05
 */
public class DukeWorld extends World
{
    private String[] keyNames =
        {"a", "s", "d", "f"};
    private String[] soundNames =
        {"test.wav", "test1.wav", "test2.wav", "test3.wav"};
    /**
     * This constructor creates a new world and the objects that start the
     * game.
     */
    public DukeWorld()
    {
```

4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala

Crearea unei metode numita **makeDukes()**:

Pentru exemplificare în codificare, am mutat crearea lui Duke într-o metodă separată numită makeDukes().

```
public class DukeWorld extends World
{
    private String[] keyNames =
        {"a", "s", "d", "f"};
    private String[] soundNames =
        {"test.wav", "test1.wav", "test2.wav", "test3.wav"};
    /**
     * This constructor creates a new world and the objects that start the
     * game.
     */
    public DukeWorld()
    {
        super(600, 400, 1);
    }
    /**
     * This constructor creates Duke and assigns the keys and sound file
     */
    private void makeDukes()
    {
        int i = 0;
        while (i < 4)
        {
            Duke duke = new Duke(keyNames[i], soundNames[i]);
            addObject (duke, i*150 + 30, 100);
            i = i + 1;
        }
    }
}
```

Obiectul **Duke** atribuit unei variabile locale numită duke

Variabila i accesează toate numele tastelor și fișierelor de tip sunet în obiectul vector, în ordinea introdusă.

Instructiuni. Variabile. Siruri de caractere

1. Crearea unei bucle *while* într-un constructor pentru a construi o lume virtuala
2. Descrierea functionarii unei bucle infinite
3. Crearea unei expresii prin intermediul operatorilor logici
4. Utilizarea unei vectori pentru a stoca mai multe variabile utilizate pentru a crea o lume virtuala
5. Folosirea variabilelor de tip șir de caractere pentru a stoca și concatena şiruri de caractere

5. Folosirea variabilelor de tip șir de caractere pentru a stoca și concatena şiruri de caractere

Concatenarea sirurilor:

- Pentru a reduce numărul de caractere redundante sau expresii pe care trebuie să le scrieți în fiecare vector se poate utiliza concatenarea de șiruri de caractere.
- Un simbol plus (+) între două șiruri de caractere le pune laolaltă într-un singur șir de caractere.

Concatenarea sirurilor de caractere combina doua șiruri de caractere într-unul singur.
Concatenarea sirurilor de caractere este reprezentată prin simbolul plus (+).

5. Folosirea variabilelor de tip șir de caractere pentru a stoca și concatena şiruri de caractere

Exemplu de concatenare de șiruri de caractere:

- În loc de a introduce sirul de caractere ".wav" după fiecare nume de fișier sunet din vector, se adaugă + ".wav" după valoarea soundName in constructor.

- Numele memorat în vector ("test"), combinat cu ".wav" are ca efect numele de fișier complet pe care programul îl înțelege, "test.wav".

```
Duke duke = new Duke(keyNames[i], soundNames[i] + “.wav”);
```

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

[5] Documentatie Oracle Academy: https://www.oracle.com/webfolder/technetwork/tutorials/OracleAcademy/GreenfootSelfStudyV1/obe.html

Întrebări?