Programare orientată pe obiecte

#5 JAVA
Greenfoot (partea a IV-a)

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

Curs 5

GREENFOOT.
Constructori

5. Constructori

1. Construirea unui obiectiv al lumii folosind metoda constructor
2. Crearea unui obiectiv folosind un constructor
3. Scrierea programelor folosind noul cuvant cheie
4. Definirea scopului si sintaxei unei variabile
5. Recunoasterea unei sintaxe si a variabilelor de test
6. Scrierea programelor pentru a schimba 2 imagini
7. Scrierea programelor pentru a termina jocul

Constructori

- Cand o noua subclasa a lumii este creata si compilata, Greenfoot executa un constructor care creeaza o instanta a acestuia pentru a o afisa in scenariu
- Constructorii furnizeaza instante si le aseaza in scenariul initial, cum ar fi marimea si rezolutia exemplului:
  - *constructorii nu returneaza date*
  - *numele lor, imediat urmat de cuvantul "public" sunt aceeasi clasa cu cea in care sunt definite*

Constructorii sunt metode speciale care sunt executate automat oricand o noua instanta a unei clase este creata.

Parametrii constructor:

- *Un parametru constructor permite unei valori initiale a instantei sa fie trecuta prin constructor.*

Acesti parametri:

- sunt disponibili instantei create de constructor
- au un scop restrictiv limitat cand este facuta declararea constructorului
- au o durata de functionare limitata de unica executare a constructorului
- dispar odata ce constructorul a terminat de executat
- sunt disponibili atata timp cat exista o instanta

Exemplu de constructor

- Acest constructor este intr-o subclasa a World folosind cuvantul “super()” pentru a transmite valorile inaltimii, latimii si rezolutiei catre instanta:

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

/**
 * Write a description of class DukeWorld here.
 *
 * @author Oracle Academy
 * @version 1.0
 */
public class DukeWorld extends World
{
    /**
     * Constructor for objects of class DukeWorld.
     *
     */
    public DukeWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(600, 400, 1);
    }
}
```

Exemplu de parametrii

- Pentru a schimba marimea tablei de joc, modificam argumentul din parametrii constructorului.
- Acest exemplu face dimensiunea jocului ca si patrat in loc de dreptunghi schimband limita coordonatei x la 400

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

/**
 * Write a description of class DukeWorld here.
 *
 * @author Oracle Academy
 * @version 1.0
 */
public class DukeWorld extends World
{

    /**
     * Constructor for objects of class DukeWorld.
     *
     */
    public DukeWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(400, 400, 1);
    }
}
```

Crearea automata a instantelor executante

- Scrieti codul in constructorul lumii pentru a adauga automat o instanta **Actor** la joc cand scenariul este initializat

- Aceasta elimina necesitatea jucatorului de a adauga o instanta inainte ca jocul sa inceapa

- De exemplu, intr-un joc in care trebuie sa gasesti perechea, cartile ar trebui sa apara automat in scenariu atunci cand jocul incepe

Codul pentru crearea automata a unei instante:

Codul in constructorul World contine urmatoarele elemente:

- comanda super() cu dimensiunea lumii ca si argumente
- metoda addObject cu urmatoarele argumente:

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

Instantele Actor ale Greenfoot

- *Alternand intre 2 imagini care arata putin diferit ofera instantei senzatia de miscare*

Instantele **Actor** ale Greenfoot:

- primesc si retin imagini din clasele lor
  - imaginea a fost alocata clasei cand aceasta a fost creata
- are abilitatea de a tine mai multe imagini
- poate fi programata sa schimbe imaginea pe care o arata in orice moment

Clasa GreenfootImage

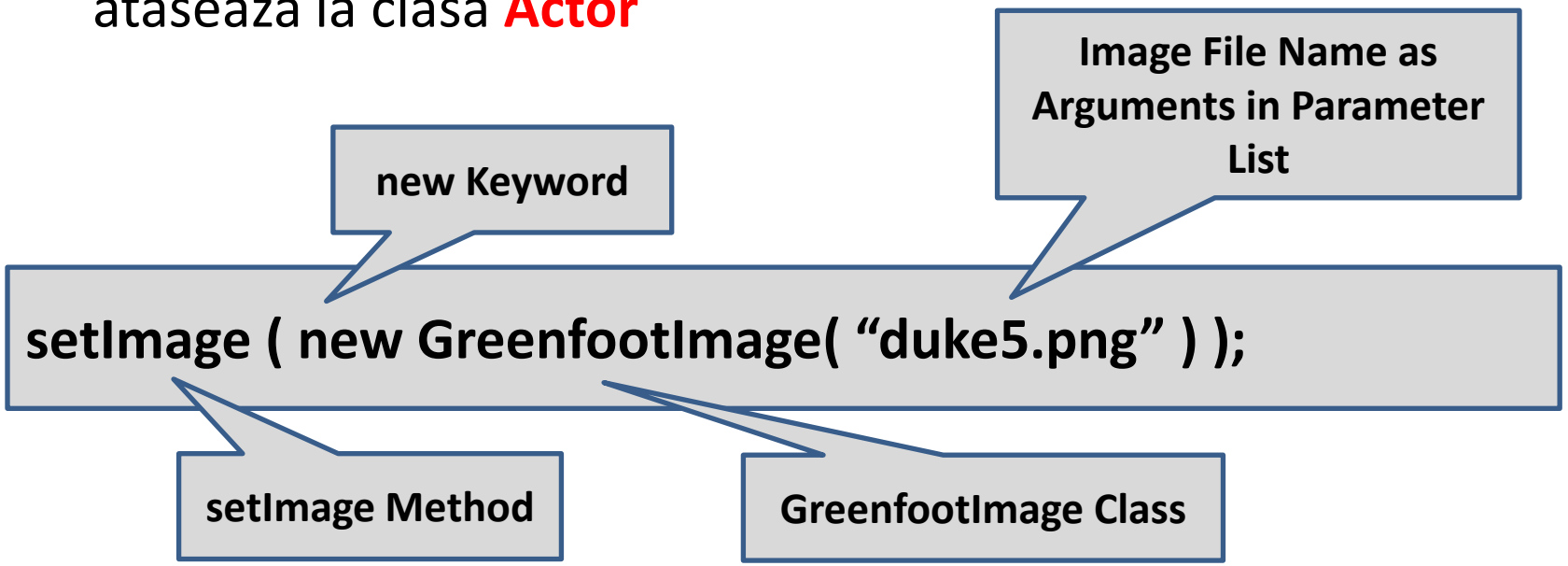
- Clasa GreenfootImage permite claselor de tip **Actor** din **Greenfoot** sa mentina vizibila imaginea acestora prin atasarea unui obiect de tip GreenfootImage

- Aceasta clasa este folosita pentru a ajuta o clasa sa obtina si sa manipuleze mai multe tipuri de imagini

- Imaginile folosite in aceasta clasa trebuie sa existe dinainte in scenariu in folderul Images

Constructor pentru obtine o noua imagine obiect

- Crearea unui constructor care extrage o noua imagine obiect dintr-un fisier atunci cand cream instanta unei clase
- Exemplul de constructor de mai jos creaza o noua imagine si o ataseaza la clasa **Actor**



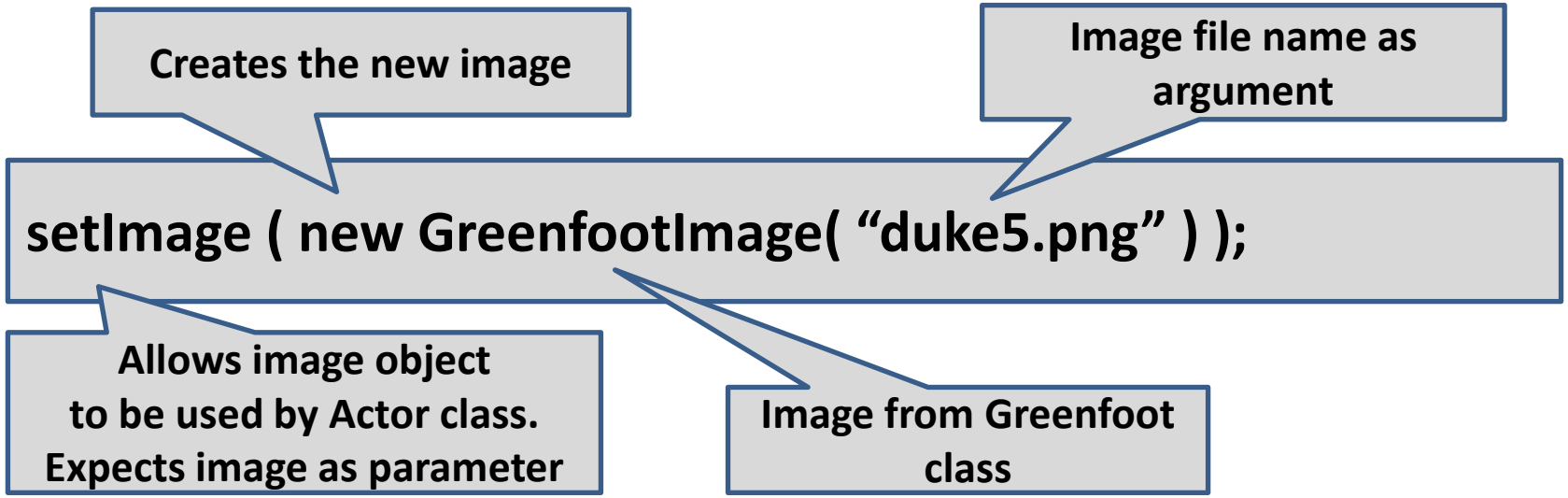
Atribuirea unei imagini noi unei clase

- *Instructiunea de mai jos creaza o noua imagine obiect din fisierul de tip imagine*
- Cand inseram in sursa unei clase, aceasta imagine obiect este gata de a fi folosita de catre clasa respectiva
- Instructiunea este executata ca mai jos:
  - Obiectul GreenfootImage este creat mai intai
  - metoda setImage este executata, facand o noua imagine obiect creata argument in lista de parametri

```
setImage ( new GreenfootImage( “duke5.png” ) );
```

Atribuirea unei imagini exemplu:

- Metoda setImage atribuie imaginea fisierului “duke5.png” clasei **Actor**.
- De fiecare data cand o instanta a acestei clase este adaugata scenariului, este afisata imaginea “duke5.png”



Motive pentru care instantele folosesc mai multe imagini:

Poate vrei ca instanta sa retina si sa acceseze mai multe imagini:

- sa para ca isi schimba culoarea
- sa para ca se schimba dintr-un obiect in altul. De exemplu, sa transformi in mod magic un iepure intr-o broasca testoasa care sa para ca se misca
- pentru *actiunea de deplasare* (exemple):
  - Plimbare: se schimba un obiect cu piciorul stang intins, intr-un obiect cu piciorul drept intins
  - Pentru a intoarce cartile: se schimba o carte alba intr-o carte care nu este alba

Accesarea mai multor imagini:

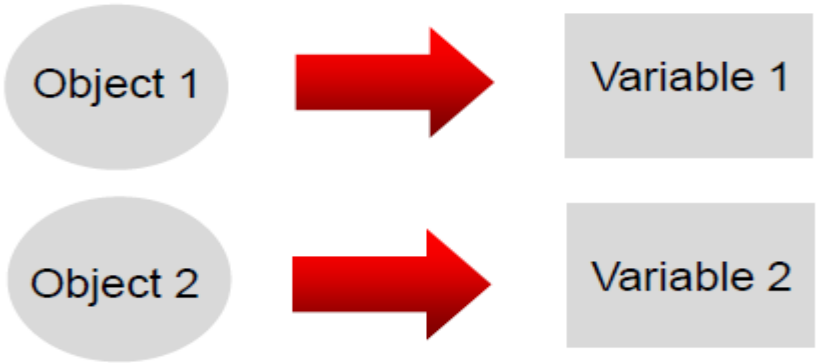
- De exemplu, o instanta poate accesa 2 imagini, fiecare cu o mica diferenta in pozitia bratelor, asa ca instanta isi misca bratul in timp ce se misca

- Pentru a avea acest gest:
  - creezi 2 imagini ale aceleasi instante, fiecare cu o pozitie diferita a bratelor
  - pui cele 2 imagini in instanta, asa incat acestea sa poata fi accesate in mod repetat ca si cand obiectul s-ar misca

Variabile

- Foloseste o variabila sa stochezi cele 2 imagini obiect in clasa.
- Aceasta permite clasei sa le acceseze mai usor cu instantele

O variabila este declarata intr-o clasa. Este folosita pentru a stoca informatiile pentru a putea fi folosite mai tarziu, sau pentru ca informatia sa fie trimisa mai departe. Se pot stoca obiecte sau valori



Formatul variabilelor:

Formatul unei variabile include:

- tipul de data: ce fel de data sa fie stocata in variabila
- numele variabilei: o descriere ce indica la ce vrea sa fie folosita variabila asa incat sa se poata referi la aceasta mai tarziu

```
private variable-type variable-name;
```

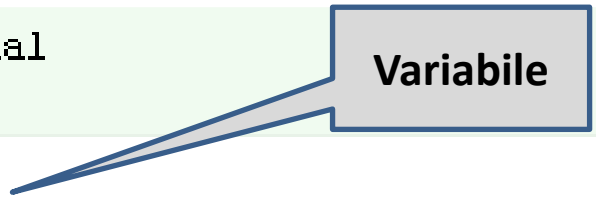
- In acest exemplu, variabila este numita image1 si tipul variabilei este GreenfootImage

```
private GreenfootImage image1;
```

Declararea variabilelor:

- Declara variabilele inainte de constuctori si metode
- Formatul pentru declararea variabilelor include:
  - cuvinte cheie care sa indice ca variabila este disponibila cu ajutorul clasei **Actor**
  - clasa careia ii apartine imaginea
  - un loc pentru variabila in care imaginea sa fie stocata

```
public class Duke extends Animal
{
private GreenfootImage image1;
private GreenfootImage image2;
```



Atribuirea instructiunilor:

- O atribuire trebuie sa stocheze obiecte intr-o variabila
- *Cand un obiect este atribuit unei variabile, variabila contine o referinta catre obiect*

O instructiune de atribuire:

- stocheaza obiectul sau valoarea intr-o variabila
- este scrisa cu simbolul "**=**"

Format:

Variable = expression;

Atribuirea componentelor instructiunilor:

O instructiune de atribuire contine:

- variabila: numele variabilei care stocheaza un obiect sau o valoare
- simbolul "=", care este simbolul atribuirii
- expresii:
  - numele obiectului sau valorii atribuite
  - o instruciune care este obiect sau valoare acum
  - clasa de care apartine imaginea

Exemple:

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

Initializarea imaginilor sau valorilor:

- *Initializarea este un proces de stabilire al instantelor si al valorilor initiale*
- Cand clasa creeaza o noua instanta, fiecare instanta contine o referire catre imagini sau valori continute in variabile

Indicatii:

- semnatura nu contine un tip ce se returneaza
- numele constructorului este acelasi ca si numele clasei
- constructorul este automat executat asftel incat sa trimita imaginea sau valoarea instantei cand instanta clasei este creata

Exemple de constructori Actor:

- Urmatorii constructori de tip **Actor** ii transmit mediului **Greenfoot** sa creeze automat o noua instanta Duke si sa initializeze, sau sa atribuie, 2 variabile acelei instante
- Ultima linie a constructorului, setImage(image1) indica ca prima variabila ar trebui sa arate cand instanta este adaugata scenariului

```
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);
    }
```

Valori de test ale variabilelor:

- Odata ce clasa a fost initializata cu 2 variabile cu imagini, programeaza instanta sa schimbe automat imaginile in timp ce se misca

- In timp ce aceste imagini sunt alternate cu fiecare miscare, face instanta sa para mai animata

- *Este posibil sa programezi schimbarea dintre imagini fara a fi nevoie sa scrii prea multe linii de cod care asociaza fiecare imagine cu fiecare miscare*

Scrierea actiunilor in pseudocod

- Identifica actiunile prin scrierea acestora in pseudocod
- Pseudocod-ul exprima activitatile sau operatiile pentru ca instanta sa faca un mix intre limbajul **Java** si cuvintele simple din engleza
- Acesta ajuta la mai buna intelegere a ce fel de comportament ar trebui instantele sa aiba inainte sa scrii codul real

Exemplu de pseudocod:

- image1 arata cand instanta este creata
- Cand Duke face urmatoarea sa miscare, image2 ar trebui sa apara, si vice-versa
- Acesta este exprimat ca si instructiunea IF-ELSE

```
if (current image displayed is image1) then
        use image2 now
else
        use image1 now
```

Operatorul “==”

Instructiunile care fac instantele sa alterneze intre imagini contin

- instructiunea IF-ELSE
- operatorul “==”

Operatorul “==”:

- este folosit intr-o instructiune IF pentru a testa cand 2 valori sunt egale
- compara o valoare cu alta
- returneaza un rezultat (adevarat sau fals)

De retinut ca simbolul de atribuire “=” nu este un simbol de egalitate

Componentele instructiunii IF-ELSE:

- Metoda getImage primeste imaginea curenta a instantei

- Operatorul **==** verifica valoarea aratata de instanta daca este egala cu image1
  - daca este egala atunci se afiseaza image2
  - altfel, se afiseaza image1

Examplu de instructiune IF-ELSE

- Instructiunea IF-ELSE de mai jos este scrisa in metoda act() pentru a afisarea alternativa a doua obiecte de tip imagine.

```
/**
 * 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()
{
    if (getImage() == image1)
    {
        setImage(image2);
    }
    else
    {
        setImage(image1);
    }
    move(2);

}
```

Terminarea jocului:

- Clasa Greenfoot are o metoda stop pe care o poti folosi la finalul jocului tau in punctul in care vrei tu

- Poate vrei ca jocul tau sa se termine cand:
  - jucatorul castiga un eveniment important
  - timpul expira
  - instanta atinge o coordonata sigura sau un obiect

Exemplu de joc cu Duke:

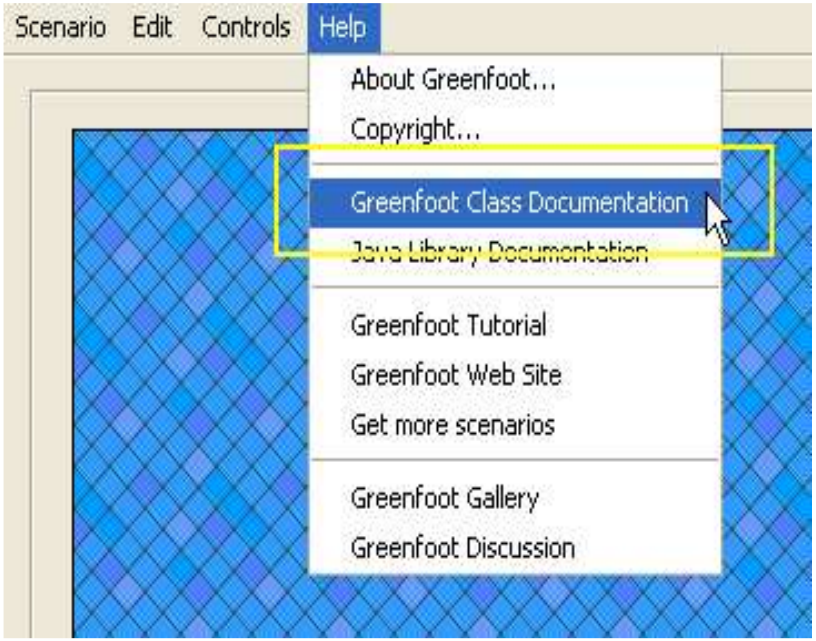
- Jucatorul decide cate obiecte Cod pot fi mancate de Duke controlat de tastatura pana la finalul jocului
- Cand jocul se termina, un sunet spune “game over”

Specificatii ale jocului:

- creaza si initializeaza variabilele pentru a manca obiecte
- numara cate obiecte au fost mancate
- permite jucatorului sa introduca numarul de obiecte pe care Duke ar trebui sa le manace cand castiga
- introdu metoda stop pentru a opri jocul cand numarul de obiecte cerute sunt mancate de Duke

Gasirea metodei stop in Greenfoot API

- Mergi la meniul help si selecteaza Greenfoot class documentation. Gaseste metoda stop in Greenfoot API



All Classes

Actor
Greenfoot
GreenfootImage
GreenfootSound
MouseInfo
World

stop

```
public static void stop()
```

Pause the execution.

Scrierea metodei stop in sursa

- In momentul in care jocul ar trebui sa se termine, scrieti metoda ca mai jos in interiorul sursei.
- Notatia punct este folosita sa apeleze metoda

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

Atribuirea variabilelor catre instante,exemple

- Duke trebuie sa manance un numar de obiecte cod definite ca jucatorul sa castige jocul.
- Variabilele sunt definite inainte de constructori si metode.
- Constructorul Duke atribuie variabilele intantelor.

```
public class Duke extends Animal
{
// counts how many code Duke has eaten.
private int codeEaten;
// the total amount of code that Duke needs to eat to win.
private int codeToWin;
// provides the balance of code Duke has eaten.
private int codeBalance;
/**
     * Create Duke and initialize codeEaten to zero.
     */
    public Duke(int codeEaten)
    {
        codeToWin = codeEaten;
        codeBalance = 0;
    }
```

Exemplu - metoda definita lookForCode

- Metoda definita lookForCode este scrisa langa metoda act() pentru a-I spune lui Duke sa se uite dupa obiecte cod.
- Daca numarul de coduri mancate este egal cu numarul de coduri necesare pentru a castiga, atunci jocul este castigat si un sunet este redat.

```
/**
 * Look for code. If Duke finds code, he eats it. Otherwise, he does nothing.
 * If Duke wins, play a sound.
 */
public void lookForCode()
{
    if (canSee(Code.class))
    {
        eat(Code.class);
        codeEaten = codeEaten + 1;
    }
    if (codeEaten == codeToWin)
    {
        Greenfoot.playSound("GameOver.wav");
        Greenfoot.stop();
    }
}
```

CURS 5 - partea II-a

Înțelegerea notiunii de **Abstractizare**

Abstractizare

- *Puteți programa o nouă instanță pentru a efectua o singură sarcină specifică, cum ar fi redarea un sunet atunci când este apăsată o tastă cheie specifică, sau afișarea unui set de întrebări și răspunsuri de fiecare dată când un joc este pornit.*
- Pentru a crea programe pe o scară mai mare, de exemplu, unul care creează 10 obiecte dintre care fiecare efectuează diferite acțiuni, aveti nevoie sa scrieti declaratii de programare care vă permit să creați în mod repetat, obiecte care efectuează diferite sarcini, doar prin furnizarea specificului pentru diferențe.

Exemplu de abstractizare

- De exemplu, dacă aveți de gând să creați 10 obiecte prin program, toate plasate în locații diferite, este ineficient să scrieti 10 linii de cod pentru fiecare obiect.

- În schimb, abstractaţi codul și scrieti declaratii mai generice pentru a se ocupa de crearea și poziționarea obiectelor.

Principiul abstractizarii

- Abstracția are scopul de a reduce dublarea de informații într-un program prin utilizarea de abstracțiuni.
- Principiul de abstractizare poate fi un gând general, cum ar fi "nu te repeta."
- De exemplu, doriți să creați o tabla de joc care are blocuri, copaci, bastoane.
  - Nu trebuie sa scrieti declaratii de programare repetitive pentru a adăuga fiecare din aceste elemente.
  - In schimb, puteți abstractiza procedura pentru a adăuga pur și simplu obiecte la o tabla de joc într-o anumită locație.

Exemplu de abstractizare in pseudocod

- Spre exemplu, veti afisa o imagine Duke si atunci cand este apelata, *fie isi va ridica mana sus inspre cer, in lateral*, *sau o va lasa jos spre pamant*.
- Codul dumneavoastra va afisa Duke si va specifica directia de afisare.
- Creati un nou Duke (indreptat spre stanga, pozitia x=6, y=20, z0)
- Creati un nou Duke (indreptat in sus, pozitia x=34, y=52, z0)
- Creati un nou Duke (indreptat in jos, pozitia x=58, y=7, z0)

Exemplu de abstractizare in pseudocod

- Imaginati-va codul necesar pentru 300 de imagini Duke.
  - Pentru a implementa abstractizarea, creati o procedura ce creaza un nou obiect care este pozitionat unde este necesar si afiseaza o imagine potrivita.
    - Accesati Procedura Obiect_nou (imagine, pozitie)

Tehnici de abstractizare

- **Abstractizarea** se foloseste in mai multe moduri in programare:

1. *O prima tehnica este aceea de a abstractiza codul de programare utilizand variabile si parametri pentru a trece diferite tipuri de informatie la o instructiune.*
2. *O alta tehnica este aceea de a identifica instructiuni de programare similare in diferite parti ale programului care pot fi implementate doar intr-un loc abstractizand diferite parti.*

Exemplu de tehnici de abstractizare

- Spre exemplu, intr-un joc de urmarire de puncte, s-ar putea sa aveti o actiune de scadere de puncte de la un total de executare si intr-o alta sectiune de joc s-ar putea sa aveti valori scazute de puncte intr-un timp scurs dintr-un total de functionare.

- Ati putea folosi abstractizarea pentru a avea un eveniment de scadere de puncte prin specificarea tipului de eveniment si multimea de scaderi de la totalul de functionare.

Constructor folosind variabile

- In acest exemplu, constructorul Duke are variabilele definite pentru a stoca cheia si valorile de sunet

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

/**
 * Write a description of class Duke here.
 *
 * @author Oracle Academy
 * @version JF_S03_L05
 */
public class Duke extends Animal
{
    private GreenfootImage image1;
    private GreenfootImage image2;
    private boolean isKeyDown;
    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("duke2.png");
        setImage(image1);
    }
```

Programarea pentru a plasa instante

- Dupa ce sunetul si variabilele chei sunt definite intr-un constructor, scrieti codul de programare pentru a adauga automat instante a clasei in lume
- Urmatoarea instanta de programare adaugata clasei Duke:
  - Creati o noua instanta a Duke de fiecare data cand DukeWorld este reinitializata, cu o cheie specifica si un fisier de sunet.
  - Plasati instanta in DukeWorld la coordonatele x si y specificate

```
addObject (new Duke (“k”, “test.wav”), 150, 150);
```

Exemplu de Constructor

- Verifica adaugare obiectului in instructiunea constructor DukeWorld.

```
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);
        addObject (new Duke ("k", "test.wav"), 150, 100);
    }
}
```

Abstractizarea codului intr-o metoda

- Puteti anticipa abstractizarea pe durata fazei de conceptie a unui proiect, sau puteti examina codul de programare la instantele identificate ce ar putea beneficia de abstractizare.

- Deseori veti recunoaste o oportunitate de a abstratiza instante de programare atunci cand liniile de scriere a codului apar in mod repetitiv.

Exemplu de abstractizare de cod la o metoda

- Verificati codul de mai jos:

```
public class ZombieWorld extends world
{
        /**
                Constructor creates new world and initial game objects
        */
public ZombieWorld()
{
                super(600, 400, 1);
                addObject (new Zombie (“1”, “1.wav”), 1, 10);
                addObject (new Zombie (“2”, “2.wav”), 2, 10);
                addObject (new Zombie (“3”, “3.wav”), 3, 10);
                addObject (new Zombie (“4”, “4.wav”), 4, 10);
                addObject (new Zombie (“5”, “5.wav”), 5, 10);
                addObject (new Zombie (“6”, “6.wav”), 6, 10);
                addObject (new Zombie (“7”, “7.wav”), 7, 10);
                addObject (new Zombie (“8”, “8.wav”), 8, 10);
                addObject (new Zombie (“9”, “9.wav”), 9, 10);
}}
```

Exemplu de abstractizare de cod la o metoda

```
public class ZombieWorld extends world
{
        /**
                Declare constructor variables
        */
        public string varKey;
        public string varSound;
        public int varX;
        int varY=10;
        /**
                Constructor creates new world and initial game objects
        */
        public ZombieWorld()
        {
                super(600, 400, 1);
                for (int i=1; i<10; i++) {
                addObject (new Zombie (varKey=i, varSound=i+".wav"), i+varX, varY);)
        }
}
```

Abstractizare simpla a codului

- Ce ar fi daca ati avea putine informatii despre Greenfoot sau programare, si ati vrea sa creati un joc care sa mute o masina(car) in jurul ecranului?
- Ati putea simplifica modul de a muta un obiect Actor in jurul ecranului prin crearea unui set simplu de metode de miscare: moveRight, moveLeft, moveUp si moveDown.
- Aceasta ofera o abstractizare mai simpla decat cea standard Greenfoot API cu ale sale metode built-in setLocation si getX/getY.

Crearea unei subclase

- Creati o subclasa a clasei Actor numita Movement care ar permite jucatorului sa ii spuna masinii sa se mute in directia dorita.
- Adaugati codul urmator pentru Movement care va seta multimea miscarilor de fiecare data cand miscarea este ceruta.

```
import greenfoot.*;
// An actor superclass that provides movement in four directions.

public class Movements extends Actor {
        private static final int speed = 4;
}
```

Crearea Metodelor de Miscare pentru Subclasa

- Apoi, adaugati urmatoarele metode de miscare pentru a realiza miscarea.
- Aceste metode simplifica si abstractizarea Greenfoot API ale metodelor getX/getY.

```
public void moveRight() {
        setLocation ( getX() + speed, getY() );
}
public void moveLeft() {
        setLocation ( getX() - speed, getY() );
}
public void moveUp() {
        setLocation ( getX(), getY() - speed );
}
public void moveDown() {
        setLocation ( getX(), getY() + speed );
}
```

Codul sursa al metodei Act

- Scrieti metoda act() a clasei Car astfel incat instantele sale sa se mute atunci cand tastele sageti sunt apasate.
- Aceasta abstractizare ascunde si automatizeaza codul mai complex, doar aratand moveLeft, moveRight, etc.

```
public void act()
{
    if (Greenfoot.isKeyDown("left") )
    {
        moveLeft();
    }
    if (Greenfoot.isKeyDown("right") )
    {
        moveRight();
    }
}
```

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?