

DOCUMENTATIA PROIECTULUI PENTRU MATERIA

PROGRAMARE ORIENTATA PE OBIECT

**SNAKE GAME**

Botanel Mirona, grupa 30224

-Proiectul reprezintă implementarea unui joc clasic de șarpe în Java, utilizând interfața grafică Swing. Jocul permite controlul unui șarpe care trebuie să mănânce mărul pentru a-și crește lungimea și scorul. Scopul jocului este să obții cât mai multe puncte evitând coliziunile cu marginea ecranului și propriul corp.

**1) CLASE:**

* ***GamePlay:***

-Această clasă extinde JPanel și implementează interfețele KeyListener și ActionListener pentru gestionarea evenimentelor de la tastatură și timer.

-Atribute:

1. *snakex* și *snakey*: Arrays pentru stocarea coordonatelor segmentelor șarpelui.
2. *left*, *right*, *up*, *down*: Variabile booleane care indică direcția de deplasare a șarpelui.
3. *dreapta*, *stanga*, *sus*, *jos*, *snakebody*, *titleImage*, *mancare*: Obiecte ImageIcon pentru diferite părți ale jocului.
4. *timp*: Obiect Timer pentru gestionarea actualizărilor periodice ale jocului.
5. *delay*: O întârziere în milisecunde pentru timer.
6. *lungimeSarpe*: Lungimea inițială a șarpelui, *moves*: Numărul de mișcări effectuate, *score*: Scorul obținut în joc.
7. *mancarex* și *mcarey*: Arrays pentru stocarea coordonatelor mărului.
8. *random*: Obiect Random pentru generarea aleatoare a coordonatelor mărului.
9. *xpos* și *ypos*: Coordonatele curente ale mărului.
10. *gameOver*: Variabilă booleană care indică sfârșitul jocului.

-Metode:

1. *paintComponent(Graphics g):* Metodă pentru desenarea componentelor grafice ale jocului.
2. *isFoodInRestrictedArea*(int foodX, int foodY): Metodă care verifică dacă mâncarea se află într-o zonă restricționată.
3. *actionPerformed(ActionEvent e):* Metodă care gestionează acțiunile la nivel de timer, inclusiv actualizarea poziției șarpelui și gestionarea coliziunii cu mărul.
4. *keyPressed(KeyEvent e):* Metodă pentru gestionarea evenimentelor de la tastatură, inclusiv direcția de deplasare a șarpelui.
5. *keyTyped(KeyEvent e):* și keyReleased(KeyEvent e): Metode care gestionază evenimentele tastei apăsate și tastei eliberate, deși nu sunt utilizate în implementare.

-Constructor: *GamePlay*: Scop principal: stabilirea starii initiale a obiectului, aloca resursele si stabileste orice alte initiale necesare

* ***Main:***

-responsabilă pentru crearea și afișarea ferestrei jocului. Iată o explicație a codului:

1. import javax.swing.\*;: Acest import face disponibile clasele și interfețele din pachetul javax.swing, care furnizează componente pentru interfața grafică (GUI) în Java.
2. import java.awt.\*;: Acest import face disponibile clasele și interfețele din pachetul java.awt, care oferă funcționalități pentru desenare și manipularea evenimentelor AWT (Abstract Window Toolkit).
3. public class Main { ... }: Se definește clasa Main, care conține metoda main, punctul de intrare al programului.
4. public static void main(String[] args) { ... }: Aceasta este metoda main, care este apelată atunci când programul începe să ruleze. Ea creează instanța JFrame pentru fereastra jocului și instanța GamePlay pentru desenarea jocului.
5. JFrame frame = new JFrame("GamePlay Example");: Se creează un obiect JFrame cu titlul "GamePlay Example".
6. GamePlay gamePlay = new GamePlay();: Se creează un obiect GamePlay, care reprezintă zona de joc.
7. frame.add(gamePlay);: Se adaugă zona de joc (GamePlay) la fereastra principală (JFrame).
8. frame.setSize(900, 700);: Se setează dimensiunile ferestrei la 900x700 pixeli.
9. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);: Se specifică comportamentul ferestrei la închiderea acesteia. JFrame.EXIT\_ON\_CLOSE înseamnă că aplicația se va închide complet când fereastra principală este închisă.
10. frame.setVisible(true);: Se face fereastra principală vizibilă, permițând utilizatorului să vadă și să interacționeze cu ea.

In cele ce urmeaza, voi prezenta, pe scurt, codul:

public GamePlay()

Stabilirea starii initiale a obiectului GamePlay.

-ex: **dreapta = new ImageIcon("C:\\Users\\user\\Downloads\\assets\\rightmouth.png");** (am adaugat, pentru capul sarpelui, acesta cand se va deplasa spre dreapta, imaginea aferenta. Analog am facut pentrustanga, sus, jos, mancare, titlu, snakebody)

-**addKeyListener(this);** Această linie adaugă un obiect KeyListener la instanța curentă a clasei GamePlay. Un KeyListener este folosit pentru a asculta evenimentele de tastatură, cum ar fi apăsarea sau eliberarea unei taste. Prin adăugarea clasei GamePlay ca ascultător pentru evenimente de tastatură, aceasta va primi notificări atunci când sunt efectuate acțiuni pe tastatură.

-**setFocusable(true);** Această metodă face componenta (în acest caz, obiectul GamePlay) capabilă să primească focus. Focusul este important în contextul tastaturii, deoarece doar componenta care are focusul va primi evenimente de la tastatură. Prin setarea acestei proprietăți la true, se permite componentei să primească focus și să asculte evenimentele de la tastatură.

-**setFocusTraversalKeysEnabled(true)**; Această metodă activează sau dezactivează utilizarea tastelor de traversare a focusului. Atunci când este setată la true, aceasta permite utilizarea tastelor de traversare a focusului pentru a naviga între diferite componente ce pot primi focus.

-**timp = new Timer(delay, this);** //Se creează un obiect Timer care va genera evenimente la intervale regulate. Constructorul Timer primește două argumente: delay (întârzierea în milisecunde între evenimente) și this (referința către obiectul care va trata aceste evenimente, în acest caz, instanța curentă a clasei GamePlay).

-**timp.start();** se porneste timer ul, incepand sa genereze evenim la interv specificate

protected void paintComponent(Graphics g) //se deseneaza toate elem grafice ale jocului

-daca **moves == 0** (inseamna ca acum a inceput jocul), se dau coordonatele initiale ale sarpelui

-daca **!gameOver** (gameOver=false; jocul nu s-a terminat), se incepe desenarea bordurilor pentru titlu, incarcarea imaginii de titlu, bordure pentru joc, se va desena scorul, iar, in cele din urma, sarpele, luandu-se in considerare variabilele right, left, up, down, care indica directia de mers a sarpelui. Se va verifica pentru fiecare din variabile

-tot aici se verifica si conditiile prin care jucatorul poate pierde: fie sarpele isi “mananca” propria coada, fie acesta a “dat cu capul” de bordure. In ambele cazuri, gameOver va devein *true*, right, left, up, down se vor reseta la *false,* si se va desena string-ul: “GAME OVER!!!”, “Space to restart” si scorul

int isFoodInRestrictedArea(int foodX, int foodY)

-Aceasta metoda se foloseste pentru a verifica dacă mâncarea se află în zona specificată (nu unde ar putea fi GameOver)

public void actionPerformed(ActionEvent e)

-Această metodă face parte din interfața ActionListener, ceea ce înseamnă că este apelată atunci când un eveniment de acțiune are loc. În acest caz, evenimentul care declanșează această metodă este generat de un obiect Timer la fiecare interval specificat (delay). Ea face parte din logica principală a jocului Snake și gestionează actualizările periodice necesare pentru mișcarea șarpelui și interacțiunea cu mărul.

-Verificarea coliziunii cu mărul: Se verifică dacă capul șarpelui se află în aceeași poziție cu mărul (coordonatele mărului coincid cu coordonatele capului).Dacă aceasta este adevărată, înseamnă că șarpele a mâncat mărul.Lungimea șarpelui (lungimeSarpe) crește cu 1.Scorul (score) crește cu 1.Se generează noi coordonate pentru măr, asigurându-se că mărul nu apare într-o zonă restricționată (funcția isFoodInRestrictedArea este utilizată pentru a verifica acest lucru).

-Miscarea sarpelui: Dacă șarpele se mișcă spre dreapta (right este true), atunci se ajustează pozițiile elementelor șarpelui pentru a reflecta o mișcare spre dreapta. Capul se deplasează în dreapta, iar restul segmentelor își schimbă pozițiile în mod corespunzător. De asemenea, se reapelează metoda repaint() pentru a redesena componenta și a reflecta noua poziție a șarpelui.Procesul este similar pentru mișcările spre stânga, sus și jos. Dacă șarpele depășește marginile ecranului sau se ciocnește cu propria sa coadă, jocul se încheie. Se afișează un mesaj "Game Over" și se cere apăsarea tastei spațiu pentru a începe un nou joc.

public void keyPressed(KeyEvent e)

-Această metodă face parte din gestionarea evenimentelor de tastatură și asigură controlul corect al șarpelui în timpul jocului Snake.

-Verificarea stării jocului (gameOver):

\*Dacă jocul s-a încheiat (gameOver este true), atunci metoda verifică dacă tasta apăsată este tasta spațiu (KeyEvent.VK\_SPACE).Dacă tasta spațiu este apăsată, jocul este resetat pentru a începe un joc nou: gameOver devine false, lungimeSarpe este resetat la 3, moves (numărul de mișcări) este resetat la 0, score este resetat la 0.

Direcția șarpelui este inițializată spre dreapta (right = true), iar celelalte direcții sunt setate pe false. Se apelează metoda repaint() pentru a redesena componenta și a începe un joc nou.

\*În cazul în care jocul continuă (gameOver este false), Numărul de mișcări (moves) este incrementat, se verifică tasta apăsată (e.getKeyCode()) pentru a determina direcția în care trebuie să se miște șarpele. Dacă tasta este săgeată dreapta (KeyEvent.VK\_RIGHT), atunci se verifică dacă șarpele se mișcă deja spre stânga (left). Dacă nu, direcția se schimbă spre dreapta, aceeași logică este aplicată și pentru celelalte direcții (left, up, down). Aceste verificări asigură că șarpele nu poate să se întoarcă brusc înapoi și să se ciocnească cu propria coadă.