Documentatie proiect laborator
Medii si instrumente de programare
Arustei Andrei 10LF331

Proiectul de la laborator consta intr-un joc Snake cu imbunatatiri si feature-uri, ce foloseste un API provider de muzica.

Lab1. Introducere in Java (output, tipuri de valori, functii)

Pentru partea de output, am afisat in consola proiectului parti unde aveam nevoie de debug sau cand am introdus date intr-un JSON.

```java
@Override
public void loadPlaylistDataFromJson() throws IOException {
    ObjectMapper objectMapper = new ObjectMapper();

    File file = new File(CURRENT_PLAYLISTDATA_PATH);

    if (file.exists() && file.length() > 0) {
        try {
            PlaylistData playlistData = objectMapper.readValue(file,
PlaylistData.class);

            this.SetPlaylistURL(playlistData.getPlaylistUrl());
            this.SetsongURLs(playlistData.getSongUrls());
            this.SetSongNames(playlistData.getSongNames());
            this.SetImageURLs(playlistData.getAlbumCoverUrls());
            this.SetArtists(playlistData.getArtistNames());
            System.out.println("Playlist data loaded from: " +
CURRENT_PLAYLISTDATA_PATH);
        } catch (Exception e) {
            System.out.println("Error reading playlist data: " +
e.getMessage());
        }
    } else {
        System.out.println("No saved playlist data found or file is empty!");
    }
    loadImagesFromJson();
}
```

Ca tipuri de date am folosit in proiect atat tipuri de date numerice (int,double,float), cat si String-uri si bool, enum-uri, variabile statice si numeroase tipuri de date din JavaFX.

```
private Deque<Point> m_snakeBody;
private Point m_snakeHead;
private direction m_currentDirection=direction.LEFT;
private Deque<javafx.scene.image.Image> m_bodyImages;
public enum direction {
    UP, DOWN, LEFT, RIGHT;
}
```

La nivel de functii, am folosit functii statice si nestatice, cu sau fara tip returnat.

```
@Override
public boolean playCurrentTrack(double VOLUME) {
    if (m_index < 0 || m_index >= m_songURLs.size()) {
        return false;
    }

    String audioUrl = m_songURLs.get(m_index);
    if (audioUrl != null) {
        Media media = new Media(audioUrl);
        mediaPlayer = new MediaPlayer(media);
        mediaPlayer.setAutoPlay(true);
        mediaPlayer.setCycleCount(1);

        mediaPlayer.setVolume(VOLUME);

        mediaPlayer.setOnEndOfMedia(() -> {
            System.out.println("Playback finished for track at index: " +
m_index);
            m_index = (m_index + 1) % m_songURLs.size();
        });

        mediaPlayer.play();
        System.out.println("Playing track at index: " + m_index);
        return true;
    } else {
        System.out.println("No audio URL available for track at index: " +
m_index);
        return false;
    }
}
```

2. Introducere in Java (Input, for, while, switch, if)

Pentru partea de input, am citit dintr-un JSON datele melodiilor pe care le incarc dupa ce fac un request api-ului.

```java
@Override
public void loadPlaylistDataFromJson() throws IOException {
    ObjectMapper objectMapper = new ObjectMapper();

    File file = new File(CURRENT_PLAYLISTDATA_PATH);

    if (file.exists() && file.length() > 0) {
        try {
            PlaylistData playlistData = objectMapper.readValue(file,
PlaylistData.class);

            this.SetPlaylistURL(playlistData.getPlaylistUrl());
            this.SetsongURLs(playlistData.getSongUrls());
            this.SetSongNames(playlistData.getSongNames());
            this.SetImageURLs(playlistData.getAlbumCoverUrls());
            this.SetArtists(playlistData.getArtistNames());
            System.out.println("Playlist data loaded from: " +
CURRENT_PLAYLISTDATA_PATH);
        } catch (Exception e) {
            System.out.println("Error reading playlist data: " +
e.getMessage());
        }
    } else {
        System.out.println("No saved playlist data found or file is empty!");
    }
    loadImagesFromJson();
}
```

Pentru partea de instructiuni repetitive si decizionale.
Urmatoarea functie decide ce actiune sa fac sarpele la fiecare frame al jocului.

```java
@Override
public void goDirection(){
    switch (m_currentDirection){
        case UP:
            moveUp();
            break;
        case DOWN:
            moveDown();
            break;
        case LEFT:
            moveLeft();
            break;
        case RIGHT:
            moveRight();
            break;
    }
}
```

Urmatoarea functie imi gaseste o melodie valida, care are si album cover si melodie disponibile si foloseste un while.

```
@Override
public void playNextSong(double VOLUME) {
    do {
        increaseIndex();
    } while (!playCurrentTrack(VOLUME));
}
```

Urmatoarea functie foloseste si doua for-uri si un if pentru a desena un gradient de nuante de verde terenului.

```
public void drawBackground(GraphicsContext gc){
    for(int i=0;i<m_numberOfRows;i++){
        for(int j=0;j<m_numberOfColumns;j++){///start with height from 
2*cellsize to not start from top left
                if ((i + j) % 2 == 0) {
                    gc.setFill(Color.web("AAD751"));
                } else {
                    gc.setFill(Color.web("A2D149"));
                }
            gc.fillRect(i * m_cellSize, (j+2)*m_cellSize , m_cellSize, 
m_cellSize);
        }
    }
}
```

Lab3. Coletii Java

Pentru a retine corpul sarpelui am folosit un Deque<Point> (Point din java.awt), iar urmatoarea functie in care am folosit iteratori inversi pentru a muta corpul sarpelui.

```
@Override
public void moveSnake(){
    Iterator<Point> itPrevPoint = m_snakeBody.descendingIterator();
    Iterator<Point> itCurrentPoint=m_snakeBody.descendingIterator();

    itPrevPoint.next();
    while (itCurrentPoint.hasNext()) {
        Point point1 = itCurrentPoint.next();
        Point point2;
        if( itPrevPoint.hasNext()){
            point2 = itPrevPoint.next();
            point1.x=point2.x;
            point1.y=point2.y;
        }
    }

    Iterator<javafx.scene.image.Image> itPrevImage =
m_bodyImages.descendingIterator();
    Iterator<javafx.scene.image.Image>
itCurrentImage=m_bodyImages.descendingIterator();

    itPrevImage.next();
    while (itCurrentImage.hasNext()) {
        javafx.scene.image.Image image1 = itCurrentImage.next();
        javafx.scene.image.Image image2;
        if( itPrevImage.hasNext()){
            image2 = itPrevImage.next();
            image1=image2;
        }
    }
}
```

Lab4. Clase Java

In proiect am folosit sase clase : Snake, Apple, Board, Playlist, Game si PlaylistData.

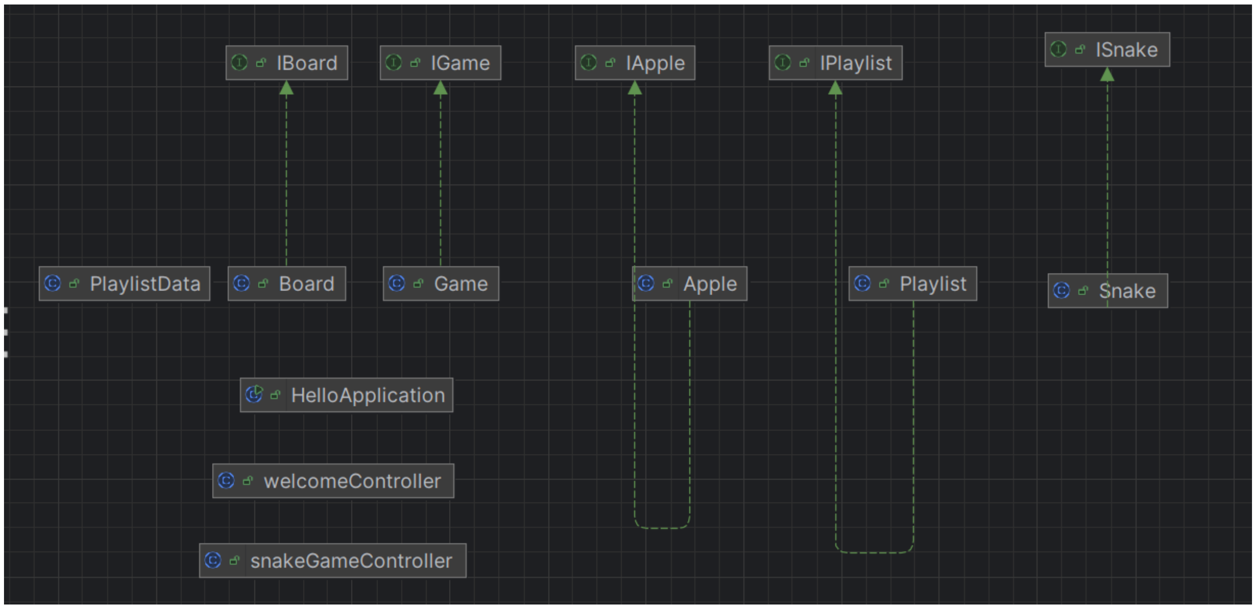
```
       Aceasta este clasa Snake
package com.example.finalsnakespotify.Models;


import com.example.finalsnakespotify.Interfaces.ISnake;
import javafx.event.ActionEvent;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.effect.Light;
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.paint.Color;
import javafx.util.Pair;

import java.awt.*;
import java.util.ArrayDeque;
import java.util.Deque;
import java.util.Iterator;
import java.util.Random;

import static com.example.finalsnakespotify.Models.Board.*;
import static com.example.finalsnakespotify.Models.Apple.*;

public class Snake implements ISnake {

    private Deque<Point> m_snakeBody;
    private Point m_snakeHead;
    private direction m_currentDirection=direction.LEFT;
    private Deque<javafx.scene.image.Image> m_bodyImages;
    public enum direction {
        UP, DOWN, LEFT, RIGHT;
```

```java
    }

    @Override
    public boolean trySpawnHead(int row,int column,int maxRows,int
maxColumns){
        return (row>=0 && row<maxRows && column>=0 && column<maxColumns);
    }

    public Snake(int rows, int columns, Image firstSongImage){
        Random rand = new Random();
        int bodyPartX = rand.nextInt(rows);
        int bodyPartY = rand.nextInt(columns-2)+2;
        m_snakeBody = new ArrayDeque<>();
        m_bodyImages = new ArrayDeque<>();
        m_snakeBody.addFirst(new Point(bodyPartX,bodyPartY));
        m_bodyImages.addFirst(firstSongImage);
        m_snakeHead=m_snakeBody.peekFirst();

        if(trySpawnHead(bodyPartX-1,bodyPartY,rows,columns))
            m_snakeBody.addLast(new Point(bodyPartX-1,bodyPartY));
        else if(trySpawnHead(bodyPartX+1,bodyPartY,rows,columns))
            m_snakeBody.addLast(new Point(bodyPartX+1,bodyPartY));
        else if(trySpawnHead(bodyPartX,bodyPartY-1,rows,columns))
            m_snakeBody.addLast(new Point(bodyPartX,bodyPartY-1));
        else if(trySpawnHead(bodyPartX,bodyPartY+1,rows,columns))
            m_snakeBody.addLast(new Point(bodyPartX,bodyPartY+1));
    }

    @Override
    public void moveSnake(){
        Iterator<Point> itPrevPoint = m_snakeBody.descendingIterator();
        Iterator<Point> itCurrentPoint=m_snakeBody.descendingIterator();

        itPrevPoint.next();
        while (itCurrentPoint.hasNext()) {
            Point point1 = itCurrentPoint.next();
            Point point2;
            if( itPrevPoint.hasNext()){
                point2 = itPrevPoint.next();
                point1.x=point2.x;
                point1.y=point2.y;
            }
        }

        Iterator<javafx.scene.image.Image> itPrevImage =
m_bodyImages.descendingIterator();
        Iterator<javafx.scene.image.Image>
itCurrentImage=m_bodyImages.descendingIterator();

        itPrevImage.next();
        while (itCurrentImage.hasNext()) {
            javafx.scene.image.Image image1 = itCurrentImage.next();
            javafx.scene.image.Image image2;
            if( itPrevImage.hasNext()){
                image2 = itPrevImage.next();
                image1=image2;
            }
        }
    }

    @Override
    public void goDirection(){
        switch (m_currentDirection){
            case UP:
                moveUp();
                break;
            case DOWN:
                moveDown();
                break;
            case LEFT:
                moveLeft();
                break;
            case RIGHT:
                moveRight();
                break;
        }
    }

    @Override
    public void drawSnake(GraphicsContext gc) {

        gc.setFill(javafx.scene.paint.Color.LIGHTBLUE);
        Point head = m_snakeHead;
        double headX = head.x * Board.GetCellSize();
        double headY = head.y * Board.GetCellSize();

        double[] xPoints = new double[3];
        double[] yPoints = new double[3];

        switch (m_currentDirection) {
            case UP:
                // Pointing upwards
                xPoints[0] = headX + Board.GetCellSize() / 2;
                yPoints[0] = headY;
                xPoints[1] = headX; // Bottom left point
                yPoints[1] = headY + Board.GetCellSize();
                xPoints[2] = headX + Board.GetCellSize();
                yPoints[2] = headY + Board.GetCellSize();
                break;
            case DOWN:
                // Pointing downwards
                xPoints[0] = headX + Board.GetCellSize() / 2;
                yPoints[0] = headY + Board.GetCellSize();
                xPoints[1] = headX; // Top left point
                yPoints[1] = headY;
                xPoints[2] = headX + Board.GetCellSize();
                yPoints[2] = headY;
                break;
            case LEFT:
                // Pointing left
                xPoints[0] = headX; // Left point
                yPoints[0] = headY + Board.GetCellSize() / 2;
                xPoints[1] = headX + Board.GetCellSize();
                yPoints[1] = headY;
                xPoints[2] = headX + Board.GetCellSize();
                yPoints[2] = headY + Board.GetCellSize();
                break;
            case RIGHT:
                // Pointing right
                xPoints[0] = headX + Board.GetCellSize();
                yPoints[0] = headY + Board.GetCellSize() / 2;
                xPoints[1] = headX;
                yPoints[1] = headY;
                xPoints[2] = headX;
                yPoints[2] = headY + Board.GetCellSize();
                break;
        }

        gc.fillPolygon(xPoints, yPoints, 3);

        Iterator<Point> iteratorVals = m_snakeBody.iterator();
        Iterator<javafx.scene.image.Image> iteratorImages =
m_bodyImages.iterator();
        Point tail = m_snakeBody.getLast();
        if (iteratorVals.hasNext() && iteratorImages.hasNext()) {
            iteratorVals.next();
            iteratorImages.next();
        }
        int index=0;
        while (iteratorVals.hasNext() && iteratorImages.hasNext()) {
            Point snakeBodyCell = iteratorVals.next();
            javafx.scene.image.Image snakeBodyImage = iteratorImages.next();

            gc.drawImage(snakeBodyImage,

snakeBodyCell.x*Board.GetCellSize(),snakeBodyCell.y*Board.GetCellSize(),
                       Board.GetCellSize(),Board.GetCellSize());
        }

        if ((tail.x + tail.y+2) % 2 == 0) {
            gc.setFill(javafx.scene.paint.Color.web("AAD751"));
        } else {
            gc.setFill(Color.web("A2D149"));
        }

        gc.fillRect(tail.x * Board.GetCellSize(), tail.y *
Board.GetCellSize(), Board.GetCellSize(), Board.GetCellSize());
    }

    @Override
    public void moveRight() {
        m_snakeHead.x+=1;
    }

    @Override
    public void moveLeft() {
        m_snakeHead.x-=1;
    }

    @Override
    public void moveUp() {
        m_snakeHead.y-=1;
    }

    @Override
    public void moveDown() {
        m_snakeHead.y+=1;
    }
    @Override
    public Deque<Point> GetSnakeBody(){
        return m_snakeBody;
    }
    @Override
    public void SetSnakeBody(Deque<Point> s){
        m_snakeBody=s;
    }
    @Override
    public direction GetCurrentDirection(){
        return m_currentDirection;
    }
    @Override
    public void SetCurrentDirection(direction currentDirection){
        m_currentDirection = currentDirection;
    }
    @Override
    public Point GetSnakeHead(){
        return m_snakeHead;
    }
    @Override
    public Deque<javafx.scene.image.Image> GetBodyImages(){return
m_bodyImages;}
    @Override
    public void SetBodyImages(Deque<javafx.scene.image.Image>
s){m_bodyImages=s;}
    @Override
    public void SetSnakeHead(int x,int y){
        m_snakeHead.x=x;
        m_snakeHead.y=y;
    }
}
```

Aceasta clasa contine un constructor al sarpelui, functia trySpawnHead care incearca la inceputul jocului sa gaseasca o pozitie valida pe teren pentru a spawna capul sau, functia moveSnake care muta sarpele la fiecare frame al jocului, functia goDirection care muta pozitia capului in functie de directia curenta, functia drawSnake care deseneaza capul din triunghuiri si corpul sarpelui din album cover-uri. Functiile moveUp,moveDown,moveLeft,moveRight muta capul sarpelui si mai avem getteri si setter pentru membrii clasei.

Lab5. Mostenire in Java, clase abstracte

In acest proiect, clasele derivate mostenesc interfete. Folosind interfete, acestea deja sunt clase abstracte prin definitie pentru ca contin doar functii pur virtuale.

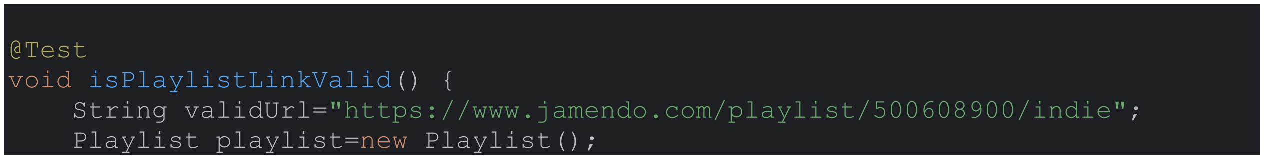


Lab6. Interfete in Java

Am folosit interfete cu scopul de a reprezenta clase de baza pentru clasele derivate, unde se afla implementarile.

Lab7. Teste in Java

Nu am realizat teste pentru toate functiile, ci doar pentru functiile pe care le-am considerat ca aplicatia nu ar rula fara. Acest test verifica daca url-ul user-ului este valid si daca se poate extrage id-ul playlist-ului.



```
@Test
void isPlaylistLinkValid() {
    String validUrl="https://www.jamendo.com/playlist/500608900/indie";
    Playlist playlist=new Playlist();
    assertEquals(true,Playlist.isPlaylistLinkValid(validUrl));
}

@Test
void extractPlaylistIdFromUrl() {
    String validUrl="https://www.jamendo.com/playlist/500608900/indie";
    Playlist playlist=new Playlist();
    assertEquals("500608900",playlist.extractPlaylistIdFromUrl(validUrl));
}
```

Lab8. Persistenta datelor

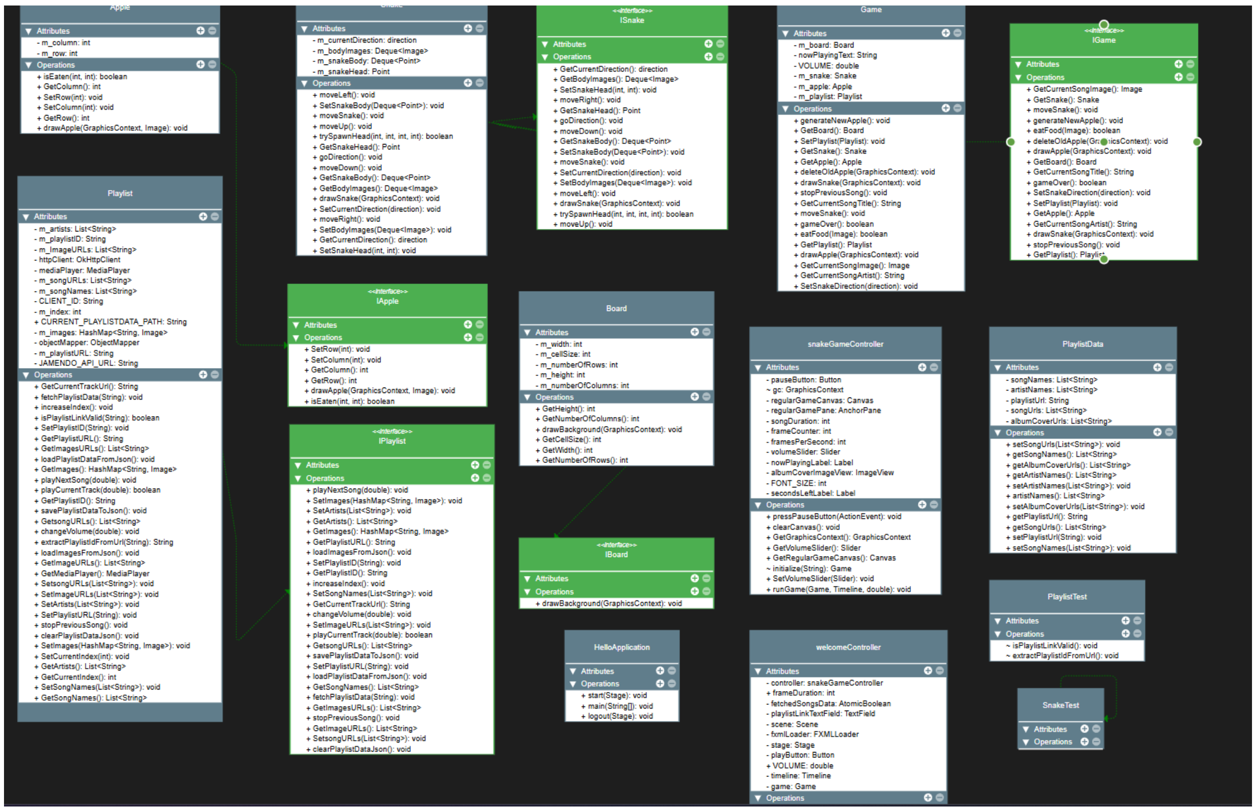
Jocul incarca datele despre melodii din request intr-un JSON, pentru a nu face mai multe request-uri ce dureaza mai mult sa fie procesate. Astfel, cand jocul este resetat dupa ce jucatorul pierde , se citesc datele din JSON-ul in care s-au salvat informatiile.

```
@Override
public void clearPlaylistDataJson() throws IOException {
    ObjectMapper objectMapper=new ObjectMapper();
    File file =new File(CURRENT_PLAYLISTDATA_PATH);
    if(file.exists()) {
        objectMapper.writeValue(file, new PlaylistData());
    }
}

@Override
public void savePlaylistDataToJson() throws IOException {
    clearPlaylistDataJson();
    ObjectMapper objectMapper = new ObjectMapper();
    PlaylistData playlistData = new PlaylistData();

    playlistData.setPlaylistUrl(this.GetPlaylistURL());
    playlistData.setSongUrls(this.GetsongURLs());
    playlistData.setSongNames(this.GetSongNames());
    playlistData.setAlbumCoverUrls(this.GetImageURLs());
    playlistData.setArtistNames(this.GetArtists());

    objectMapper.writeValue(new File(CURRENT_PLAYLISTDATA_PATH),
playlistData);
}

@Override
public void loadPlaylistDataFromJson() throws IOException {
    ObjectMapper objectMapper = new ObjectMapper();

    File file = new File(CURRENT_PLAYLISTDATA_PATH);

    if (file.exists() && file.length() > 0) {
        try {
            PlaylistData playlistData = objectMapper.readValue(file,
PlaylistData.class);

            this.SetPlaylistURL(playlistData.getPlaylistUrl());
            this.SetsongURLs(playlistData.getSongUrls());
            this.SetSongNames(playlistData.getSongNames());
            this.SetImageURLs(playlistData.getAlbumCoverUrls());
            this.SetArtists(playlistData.getArtistNames());
            System.out.println("Playlist data loaded from: " +
CURRENT_PLAYLISTDATA_PATH);
        } catch (Exception e) {
            System.out.println("Error reading playlist data: " +
e.getMessage());
        }
    } else {
        System.out.println("No saved playlist data found or file is empty!");
    }
    loadImagesFromJson();
}
```

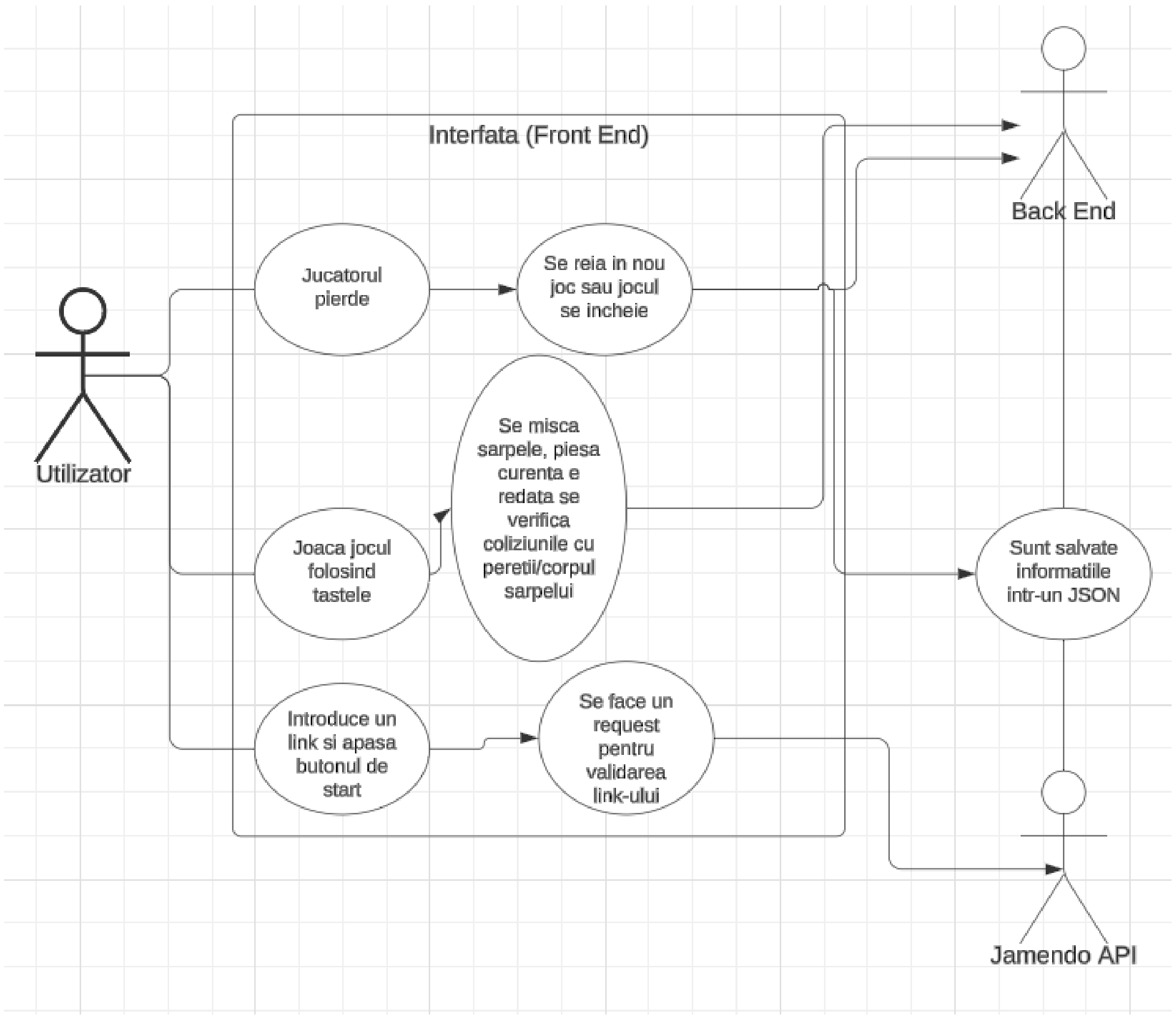
Lab9. Diagrame UML

Am realizat diagrame UML pentru toate clasele folosite in proiect.

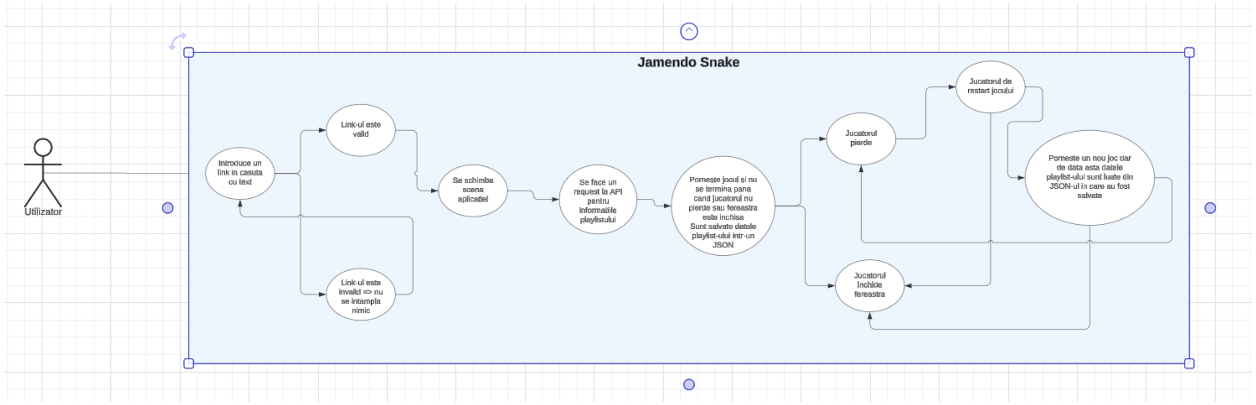
1.Class diagram


Apple
Attributes
- m_column: int
- m_row: int
Operations
+ isEaten(int, int): boolean
+ GetColumn(): int
+ SetRow(int): void
+ SetColumn(int): void
+ GetRow(): int
+ drawApple(GraphicsContext, Image): void
Attributes
- m_currentDirection: direction
- m_bodyImages: Deque<Image>
- m_snakeBody: Deque<Point>
- m_snakeHead: Point
Operations
+ moveLeft(): void
+ SetSnakeBody(Deque<Point>): void
+ moveSnake(): void
+ moveUp(): void
+ trySpawnHead(int, int, int, int): boolean
+ GetSnakeHead(): Point
+ goDirection(): void
+ moveDown(): void
+ GetSnakeBody(): Deque<Point>
+ GetBodyImages(): Deque<Image>
+ drawSnake(GraphicsContext): void
+ SetCurrentDirection(direction): void
+ moveRight(): void
+ SetBodyImages(Deque<Image>): void
+ GetCurrentDirection(): direction
+ SetSnakeHead(int, int): void
<<Interface>>
ISnake
Attributes
Operations
+ GetCurrentDirection(): direction
+ GetBodyImages(): Deque<Image>
+ SetSnakeHead(int, int): void
+ moveRight(): void
+ GetSnakeHead(): Point
+ goDirection(): void
+ moveDown(): void
+ GetSnakeBody(): Deque<Point>
+ SetSnakeBody(Deque<Point>): void
+ moveSnake(): void
+ SetCurrentDirection(direction): void
+ SetBodyImages(Deque<Image>): void
+ moveLeft(): void
+ drawSnake(GraphicsContext): void
+ trySpawnHead(int, int, int, int): boolean
+ moveUp(): void
Game
Attributes
- m_board: Board
- nowPlayingText: String
- VOLUME: double
- m_snake: Snake
- m_apple: Apple
- m_playlist: Playlist
Operations
+ generateNewApple(): void
+ GetBoard(): Board
+ SetPlaylist(Playlist): void
+ GetSnake(): Snake
+ GetApple(): Apple
+ deleteOldApple(GraphicsContext): void
+ drawSnake(GraphicsContext): void
+ stopPreviousSong(): void
+ GetCurrentSongTitle(): String
+ moveSnake(): void
+ gameOver(): boolean
+ eatFood(Image): boolean
+ GetPlaylist(): Playlist
+ drawApple(GraphicsContext): void
+ GetCurrentSongImage(): Image
+ GetCurrentSongArtist(): String
+ SetSnakeDirection(direction): void
<<Interface>>
IGame
Attributes
Operations
+ GetCurrentSongImage(): Image
+ GetSnake(): Snake
+ moveSnake(): void
+ generateNewApple(): void
+ eatFood(Image): boolean
+ deleteOldApple(GraphicsContext): void
+ drawApple(GraphicsContext): void
+ GetBoard(): Board
+ GetCurrentSongTitle(): String
+ gameOver(): boolean
+ SetSnakeDirection(direction): void
+ SetPlaylist(Playlist): void
+ GetApple(): Apple
+ GetCurrentSongArtist(): String
+ drawSnake(GraphicsContext): void
+ stopPreviousSong(): void
+ GetPlaylist(): Playlist
Playlist
Attributes
- m_artists: List<String>
- m_playlistID: String
- m_ImageURLs: List<String>
- httpClient: OkHttpClient
- mediaPlayer: MediaPlayer
- m_songURLs: List<String>
- m_songNames: List<String>
- CLIENT_ID: String
- m_index: int
+ CURRENT_PLAYLISTDATA_PATH: String
- m_images: HashMap<String, Image>
- objectMapper: ObjectMapper
- m_playlistURL: String
- JAMENDO_API_URL: String
Operations
+ GetCurrentTrackUrl(): String
+ fetchPlaylistData(String): void
+ increaseIndex(): void
+ isPlaylistLinkValid(String): boolean
+ SetPlaylistID(String): void
+ GetPlaylistURL(): String
+ GetImagesURLs(): List<String>
+ loadPlaylistDataFromJson(): void
+ GetImages(): HashMap<String, Image>
+ playNextSong(double): void
+ playCurrentTrack(double): boolean
+ GetPlaylistID(): String
+ savePlaylistDataToJson(): void
+ GetsongURLs(): List<String>
+ changeVolume(double): void
+ extractPlaylistIdFromUrl(String): String
+ loadImagesFromJson(): void
+ GetImageURLs(): List<String>
+ GetMediaPlayer(): MediaPlayer
+ SetsongURLs(List<String>): void
+ SetImageURLs(List<String>): void
+ SetArtists(List<String>): void
+ SetPlaylistURL(String): void
+ stopPreviousSong(): void
+ clearPlaylistDataJson(): void
+ SetImages(HashMap<String, Image>): void
+ SetCurrentIndex(int): void
+ GetArtists(): List<String>
+ GetCurrentIndex(): int
+ SetSongNames(List<String>): void
+ GetSongNames(): List<String>
<<Interface>>
IApple
Attributes
Operations
+ SetRow(int): void
+ SetColumn(int): void
+ GetColumn(): int
+ GetRow(): int
+ drawApple(GraphicsContext, Image): void
+ isEaten(int, int): boolean
Board
Attributes
- m_width: int
- m_cellSize: int
- m_numberOfRows: int
- m_height: int
- m_numberOfColumns: int
Operations
+ GetHeight(): int
+ GetNumberOfColumns(): int
+ drawBackground(GraphicsContext): void
+ GetCellSize(): int
+ GetWidth(): int
+ GetNumberOfRows(): int
<<Interface>>
IPlaylist
Attributes
Operations
+ playNextSong(double): void
+ SetImages(HashMap<String, Image>): void
+ SetArtists(List<String>): void
+ GetArtists(): List<String>
+ GetImages(): HashMap<String, Image>
+ GetPlaylistURL(): String
+ loadImagesFromJson(): void
+ SetPlaylistID(String): void
+ GetPlaylistID(): String
+ increaseIndex(): void
+ SetSongNames(List<String>): void
+ GetCurrentTrackUrl(): String
+ changeVolume(double): void
+ SetImageURLs(List<String>): void
+ playCurrentTrack(double): boolean
+ GetsongURLs(): List<String>
+ savePlaylistDataToJson(): void
+ SetPlaylistURL(String): void
+ loadPlaylistDataFromJson(): void
+ GetSongNames(): List<String>
+ fetchPlaylistData(String): void
+ GetImagesURLs(): List<String>
+ stopPreviousSong(): void
+ GetImageURLs(): List<String>
+ SetsongURLs(List<String>): void
+ clearPlaylistDataJson(): void
<<Interface>>
IBoard
Attributes
Operations
+ drawBackground(GraphicsContext): void
HelloApplication
Attributes
Operations
+ start(Stage): void
+ main(String[]): void
+ logout(Stage): void
snakeGameController
Attributes
- pauseButton: Button
~ gc: GraphicsContext
- regularGameCanvas: Canvas
- regularGamePane: AnchorPane
- songDuration: int
- frameCounter: int
- framesPerSecond: int
- volumeSlider: Slider
- nowPlayingLabel: Label
- albumCoverImageView: ImageView
- FONT_SIZE: int
- secondsLeftLabel: Label
Operations
+ pressPauseButton(ActionEvent): void
+ clearCanvas(): void
+ GetGraphicsContext(): GraphicsContext
+ GetVolumeSlider(): Slider
+ GetRegularGameCanvas(): Canvas
~ initialize(String): Game
+ SetVolumeSlider(Slider): void
+ runGame(Game, Timeline, double): void
PlaylistData
Attributes
- songNames: List<String>
- artistNames: List<String>
- playlistUrl: String
- songUrls: List<String>
- albumCoverUrls: List<String>
Operations
+ setSongUrls(List<String>): void
+ getSongNames(): List<String>
+ getAlbumCoverUrls(): List<String>
+ getArtistNames(): List<String>
+ setArtistNames(List<String>): void
+ artistNames(): List<String>
+ setAlbumCoverUrls(List<String>): void
+ getPlaylistUrl(): String
+ getSongUrls(): List<String>
+ setPlaylistUrl(String): void
+ setSongNames(List<String>): void
PlaylistTest
Attributes
Operations
~ isPlaylistLinkValid(): void
~ extractPlaylistIdFromUrl(): void
welcomeController
Attributes
- controller: snakeGameController
+ frameDuration: int
- fetchedSongsData: AtomicBoolean
- playlistLinkTextField: TextField
- scene: Scene
- fxmlLoader: FXMLLoader
- stage: Stage
- playButton: Button
+ VOLUME: double
- timeline: Timeline
- game: Game
Operations
SnakeTest
Attributes
Operations

2. Use case diagram


Interfata (Front End)
Utilizator
Jucatorul pierde
Se reia in nou joc sau jocul se incheie
Se misca sarpele, piesa curenta e redata se verifica coliziunile cu peretii/corpul sarpelui
Joaca jocul folosind tastele
Introduce un link si apasa butonul de start
Se face un request pentru validarea link-ului
Back End
Sunt salvate informatiile intr-un JSON
Jamendo API

3. Activity/flow diagram


Jamendo Snake
Utilizator
Introduce un link in casuta cu text
Link-ul este valid
Link-ul este invalid => nu se intampla nimic
Se schimba scena aplicatiei
Se face un request la API pentru informatiile playlistului
Porneste jocul si nu se termina pana cand jucatorul nu pierde sau fereastra este inchisa Sunt salvate datele playlist-ului intr-un JSON
Jucatorul pierde
Jucatorul da restart jocului
Porneste un nou joc dar de data asta datele playlist-ului sunt luate din JSON-ul in care au fost salvate
Jucatorul inchide fereastra