*Dependency: JAVA JDK SE17*

**Blueprint/Classes**

**1. Shop Class:**

* Think of the Shop class as the heart of an online video game store.
* It uses an ArrayList, which is like a dynamic list, to keep track of the video games available for purchase.
* When you create a "Shop" object, it's preloaded with some default games.
* You can use this class to check which games the store has in stock.
* package soph\_part;
* import java.util.\*;
* public class Shop {
* private ArrayList<Game> gamesLibrary;
* public Shop() {
* gamesLibrary = new ArrayList<>();
* defaultGames();
* }
* public ArrayList<Game> getShopGamesLibrary(){
* return this.gamesLibrary;
* }
* private void defaultGames() {
* Game legend\_of\_zelda = new Game("Legend of Zelda", 2500, 0);
* Game hellblade = new Game("Hellblade", 1500, 0);
* Game castlevania = new Game("Castlevania - Symphony of the night", 700, 0);
* Game mass\_effect = new Game("Mass Effect", 2500, 0);
* Game horizon\_zero\_dawn = new Game("Horizon Zero Dawn", 500, 0);
* Game spiderman = new Game("Spiderman", 2200, 0);
* Game control = new Game("Control", 1300, 0);
* Game hades = new Game("Hades", 800, 0);
* Game hollow\_night = new Game("Hollow Night", 1000, 0);
* Game bioshock = new Game("Bioshock", 2800, 0);
* Game story\_of\_season = new Game("Story of Season", 500, 0);
* Game sea\_of\_stars = new Game("Sea of Stars", 1099, 0);
* Game half\_life = new Game("Half Life", 200, 0);
* gamesLibrary.add(legend\_of\_zelda);
* gamesLibrary.add(hellblade);
* gamesLibrary.add(castlevania);
* gamesLibrary.add(mass\_effect);
* gamesLibrary.add(horizon\_zero\_dawn);
* gamesLibrary.add(spiderman);
* gamesLibrary.add(control);
* gamesLibrary.add(hades);
* gamesLibrary.add(hollow\_night);
* gamesLibrary.add(bioshock);
* gamesLibrary.add(story\_of\_season);
* gamesLibrary.add(sea\_of\_stars);
* gamesLibrary.add(half\_life);
* }
* }

**2. User Class:**

* The User class represents a customer of the online game store.
* It stores important user information like their username, password, and a list of games they own.
* The list of games is managed as a stack, which means you can add games to the top (like stacking plates) and remove games from the top (like taking off the top plate).
* The User class also keeps track of the user's game achievement score.
* package soph\_part;
* import java.util.\*;
* public class User {
* private String username;
* private String password;
* private Stack<Game> gamesLibrary;
* private int gameAchievement;
* public User(String username, String password) {
* this.username = username;
* this.password = password;
* this.gamesLibrary = new Stack<>();
* this.gameAchievement = 0;
* }
* // Getters and Setters for User class
* public String getUsername() {
* return username;
* }
* public String getPassword() {
* return password;
* }
* public Stack<Game> getGamesLibrary() {
* return gamesLibrary;
* }
* public int getGameAchievement() {
* return gameAchievement;
* }
* // Method to add a game to the user's library
* public void addGameToLibrary(Game game) {
* gamesLibrary.push(game);
* }
* // Method to remove and return the top game from the library
* public Game removeTopGame() {
* if (!gamesLibrary.isEmpty()) {
* return gamesLibrary.pop(); // Pop the top game from the stack
* } else {
* return null; // Stack is empty
* }
* }
* }

**3. Game Class:**

* The Game class focuses on the characteristics of a single video game in the store.
* It stores details such as the game's title, price, and level.
* You can create a new Game object and specify these details.
* Additionally, there's a simplified version of this class where you can create a game with just its title, and the price and level are set to default values.
* package soph\_part;
* public class Game {
* private String title;
* private double price;
* private int level;
* public Game(String title, double price, int level) {
* this.title = title;
* this.price = price;
* this.level = level;
* }
* public Game(String title) {
* this.title = title;
* this.price = 0;
* this.level = 0;
* }
* // Getters for Game class
* public String getTitle() {
* return title;
* }
* public double getPrice() {
* return price;
* }
* public int getLevel() {
* return level;
* }
* }

These classes collaborate to build an online video game store. The Shop class maintains the list of available games, the User class represents the store's customers and their game collections, and the Game class provides specific information about each game. This setup allows users to purchase, own, and manage their video games within the store.

**MAIN CLASS**

**1. User Class:**

* The "User" class represents a user of the program and stores user-related information such as their username, password, and a collection of video games.
* Users can add and remove games from their collection using this class.

**2. Shop Class:**

* The "Shop" class acts as a virtual video game store and keeps track of available video games. It also contains default games.

**3. Game Class:**

* The "Game" class describes individual video games and includes information like the game's title, price, and level.

**4. Soph Class:**

* The "Soph" class serves as the main program class.
* It creates a new user object and associates it with a specific username and password ("soph" and "soph123" in this case).

**5. main Method:**

* The program starts in the **main** method.
* An instance of the "Soph" class is created, which automatically creates a user.
* It also creates an instance of the "Shop" class, representing the virtual store.
* The current user is obtained using the **getCurrentUser** method.

**6. fetchGamesToGamesFrame Method:**

* This method creates a graphical interface to display a user's game library in a frame.
* It prepares the data from the user's library and displays it in the "GamesFrame" frame.

**7. addGamesToCurrentUserLibrary Method:**

* This method is responsible for creating and adding specific games to the user's game library. Three games are added with their titles, prices, and levels.

**8. sendUserDataToShopFrame Method:**

* This method creates a graphical interface to display the shop's content in a frame.
* It prepares the data from the user and the shop and displays it in the "ShopFrame" frame.

**9. sendUserDataToAchievementsFrame Method:**

* This method is designed to send user data to an "AchievementsFrame" frame, but it is currently commented out.

In summary, the code demonstrates the basic structure of a video game management system. It creates a user, adds games to their library, and provides ways to interact with the user's game collection and the virtual game store. The user interface for displaying games and achievements is also part of the program but currently commented out.

package soph\_part;

import java.util.\*;

import javax.swing.JLabel;

/\*\*

\*

\*/

public class Soph {

private User user; //initialize the user variable from User class in Soph class

public Soph() {

this.user = new User("soph", "soph123"); // Create the user

}

public User getCurrentUser() {

return this.user; //return the current user object

}

public static void main(String[] args) {

Soph soph = new Soph(); //Instance of this class

Shop shop\_obj = new Shop(); //Instance of Shop class

User user\_obj = soph.getCurrentUser(); // get the user object using the getter method in Soph class

soph.addGamesToCurrentUserLibrary(user\_obj); //call method to add games and pass the user\_obj as parameter(line:45)

//soph.fetchGamesToGamesFrame(user\_obj); //shows the GamesFrame Frame <-remove comment to test

*sendUserDataToShopFrame*(user\_obj, shop\_obj); //shows the ShopFrameFrame <-remove comment to test

//sendUserDataToAchievementsFrame(user\_obj); //shows the AchievementsFrame <-remove comment to test

}

private void fetchGamesToGamesFrame(User user) { //create method for fetching games to GamesFrame

GamesFrame games\_frame\_obj = new GamesFrame(user); //create instance of GamesFrame class

Stack<Game> user\_games = user.getGamesLibrary(); //Create new stack variable and pass the value of user\_object get library

//send users games data to games frame and set visible

games\_frame\_obj.updateGamesInVault(user\_games);

games\_frame\_obj.setVisible(true);

}

private void addGamesToCurrentUserLibrary(User user) {

// Create some games

Game rrd\_2 = new Game("Read Dead Redemption 2", 29.99, 1);

Game gta\_v = new Game("GTA V", 39.99, 2);

Game dota\_2 = new Game("DOTA 2", 39.99, 1);

// User adds local game

user.addGameToLibrary(rrd\_2);

user.addGameToLibrary(gta\_v);

user.addGameToLibrary(dota\_2);

}

private static void sendUserDataToShopFrame(User user, Shop shop) {

ShopFrame shop\_frame\_obj = new ShopFrame(user, shop); //create instance of ShopFrame Frame

shop\_frame\_obj.setVisible(true); //set ShopFrame visible

}

private static void sendUserDataToAchievementsFrame(User user) {

AchievementsFrame af = new AchievementsFrame(user); //create instance of AchievementsFrame Frame

af.setVisible(true); //set AchievementsFrame visible

}

}

**GUI/FRAMES**

***1. ShopFrame Class:***

* The "ShopFrame" class represents a graphical user interface for the virtual video game store.
* It extends the "JFrame" class, which is a window for a Java Swing application.

**2. Main Method:**

* The main method is the entry point for the program.
* It creates an instance of the "ShopFrame" and makes it visible.
* It also sets the current user and shop objects.

**3. Constructor:**

* The constructor of the "ShopFrame" class takes two parameters, a user, and a shop, to associate them with the frame.
* Inside the constructor, it initializes the background image and sets up the frame's properties.

**4. MainPanel Class:**

* This is a nested class within "ShopFrame."
* It extends "JPanel" and is responsible for rendering the background image on the frame.
* It overrides the **paintComponent** method to draw the background image.

**5. Components and Layout:**

* The frame contains several panels (JPanel), labels (JLabel), buttons (JButton), and a scroll pane (JScrollPane) to create the user interface.
* These components are arranged to display information about games in the virtual store and the user's game collection.

**6. Adding Games to the Shop:**

* The code creates a list of games available in the shop and displays them in a scrollable panel.
* For each game, it creates a label with the game's title and a button to allow the user to purchase the game.
* When the "Buy" button is clicked, it adds the selected game to the user's game library and removes it from the shop's library.
* The frame is then refreshed to reflect the changes.

**7. Adding User's Games to "My Vault":**

* It also displays the user's game collection in a separate panel labeled "My Vault."

**8. Scroll Panes:**

* Scroll panes are used to manage and display lists of games, making it easy to scroll through a potentially long list of games.

**9. Background Image:**

* The frame has a background image (a graphical background) to make the interface visually appealing.
* If the image fails to load, it falls back to using a regular JPanel.

In summary, the "ShopFrame" class is responsible for displaying the virtual store where users can browse and purchase games. It provides a graphical interface for viewing games in the shop and the user's own collection. When a user buys a game, the frame updates to reflect the changes in the user's library and the shop's inventory.

public class ShopFrame extends JFrame {

private BufferedImage backgroundImage;

private static final long ***serialVersionUID*** = 1L;

private JPanel contentPane;

private User current\_user;

private Shop shop\_obj;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.*invokeLater*(new Runnable() {

public void run() {

try {

ShopFrame frame = new ShopFrame(null, null);

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public ShopFrame(User user, Shop shop) {

this.current\_user = user;

this.shop\_obj = shop;

ArrayList<Game> shopGamesList = shop\_obj.getShopGamesLibrary();

ArrayList<String> userGamesList = new ArrayList<>();

try {

BufferedImage backgroundImage = ImageIO.*read*(new File("src/img/shops.jpg"));

contentPane = new MainPanel(backgroundImage);

contentPane = new MainPanel(backgroundImage);

} catch (IOException e) {

e.printStackTrace();

contentPane = new JPanel(); // Default to a regular JPanel if the image load fails

}

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setBounds(100, 100, 682, 603);

// Use a regular JPanel

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

contentPane.setOpaque(false);

JPanel panel = new JPanel();

panel.setBounds(0, 10, 668, 556);

contentPane.add(panel);

panel.setLayout(null);

panel.setOpaque(false);

JPanel panel\_1 = new JPanel();

panel\_1.setLayout(null);

panel\_1.setBounds(0, 0, 668, 140);

panel.add(panel\_1);

panel\_1.setOpaque(false);

JPanel panel\_2 = new JPanel();

panel\_2.setLayout(null);

panel\_2.setBounds(224, 24, 189, 68);

panel\_2.setOpaque(false);

panel\_1.add(panel\_2);

JLabel lblNewLabel = new JLabel("Store");

lblNewLabel.setHorizontalTextPosition(SwingConstants.***CENTER***);

lblNewLabel.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel.setFont(new Font("Perpetua Titling MT", Font.***PLAIN***, 25));

lblNewLabel.setForeground(Color.***WHITE***);

lblNewLabel.setBounds(10, 10, 169, 48);

panel\_2.add(lblNewLabel);

JButton btnNewButton\_3 = new JButton("Back");

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose();

}

});

btnNewButton\_3.setBounds(10, 10, 91, 36);

panel.add(btnNewButton\_3);

JPanel panel\_3 = new JPanel();

panel\_3.setBounds(10, 150, 319, 396);

panel.add(panel\_3);

panel\_3.setLayout(null);

JPanel gamePanel = new JPanel();

gamePanel.setLayout(null);

int i = 0;

for (Game game : shopGamesList) {

JLabel newLabel = new JLabel(game.getTitle());

newLabel.setBounds(10, 10 + (i \* 30), 200, 25);

gamePanel.add(newLabel);

JButton buyButton = new JButton("Buy");

buyButton.setBounds(200, 10 + (i \* 30), 80, 25);

gamePanel.add(buyButton);

final int gameIndex = i; // Capture the current game index

buyButton.addActionListener(new ActionListener() {

*@Override*

public void actionPerformed(ActionEvent e) {

// Get the selected game from the shop

Game selectedGame = shop\_obj.getShopGamesLibrary().get(gameIndex);

// Add the selected game to the user's game library

current\_user.addGameToLibrary(selectedGame);

// Remove the purchased game from the shop's game library

shop\_obj.getShopGamesLibrary().remove(gameIndex);

// Refresh the display to reflect the changes (you may need to recreate the entire ShopFrame)

dispose();

new ShopFrame(current\_user, shop\_obj).setVisible(true);

}

});

i++;

}

gamePanel.setPreferredSize(new Dimension(289, i \* 30 + 30)); // Adjust the width and height

//scrollpane for games

JScrollPane scrollPane = new JScrollPane(gamePanel);

scrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.***HORIZONTAL\_SCROLLBAR\_NEVER***);

scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.***VERTICAL\_SCROLLBAR\_AS\_NEEDED***);

scrollPane.setBounds(10, 53, 299, 333);

panel\_3.add(scrollPane);

JLabel lblNewLabel\_1 = new JLabel("Browse from Store");

lblNewLabel\_1.setFont(new Font("Perpetua Titling MT", Font.***PLAIN***, 14));

lblNewLabel\_1.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel\_1.setBounds(10, 10, 299, 34);

panel\_3.add(lblNewLabel\_1);

JPanel panel\_3\_1 = new JPanel();

panel\_3\_1.setLayout(null);

panel\_3\_1.setBounds(339, 150, 319, 396);

panel.add(panel\_3\_1);

for (Game game : current\_user.getGamesLibrary()) {

userGamesList.add(game.getTitle());

}

JList<String> userGameList = new JList<>(userGamesList.toArray(new String[0]));

JScrollPane scrollPane\_1 = new JScrollPane(userGameList);

scrollPane\_1.setBounds(10, 52, 299, 334);

panel\_3\_1.add(scrollPane\_1);

JLabel lblNewLabel\_1\_1 = new JLabel("My Vault");

lblNewLabel\_1\_1.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel\_1\_1.setFont(new Font("Perpetua Titling MT", Font.***PLAIN***, 14));

lblNewLabel\_1\_1.setBounds(10, 10, 299, 34);

panel\_3\_1.add(lblNewLabel\_1\_1);

}

private void handleBuyButton(Game game) {

// Add the selected game to the user's library

current\_user.addGameToLibrary(game);

}

class MainPanel extends JPanel {

private BufferedImage backgroundImage;

public MainPanel(BufferedImage backgroundImage) {

this.backgroundImage = backgroundImage;

}

*@Override*

protected void paintComponent(Graphics g) {

super.paintComponent(g);

if (backgroundImage != null) {

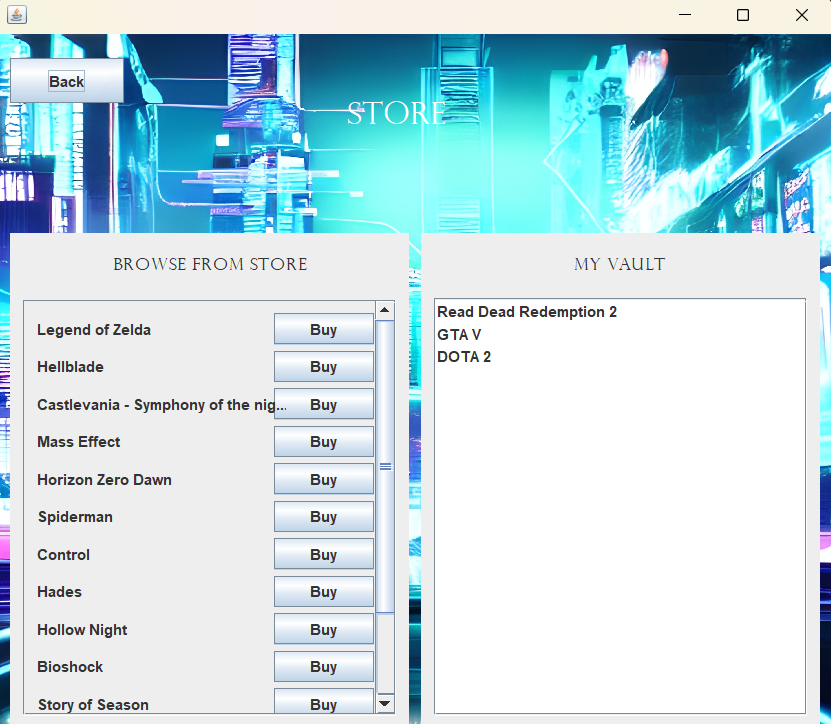
g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);

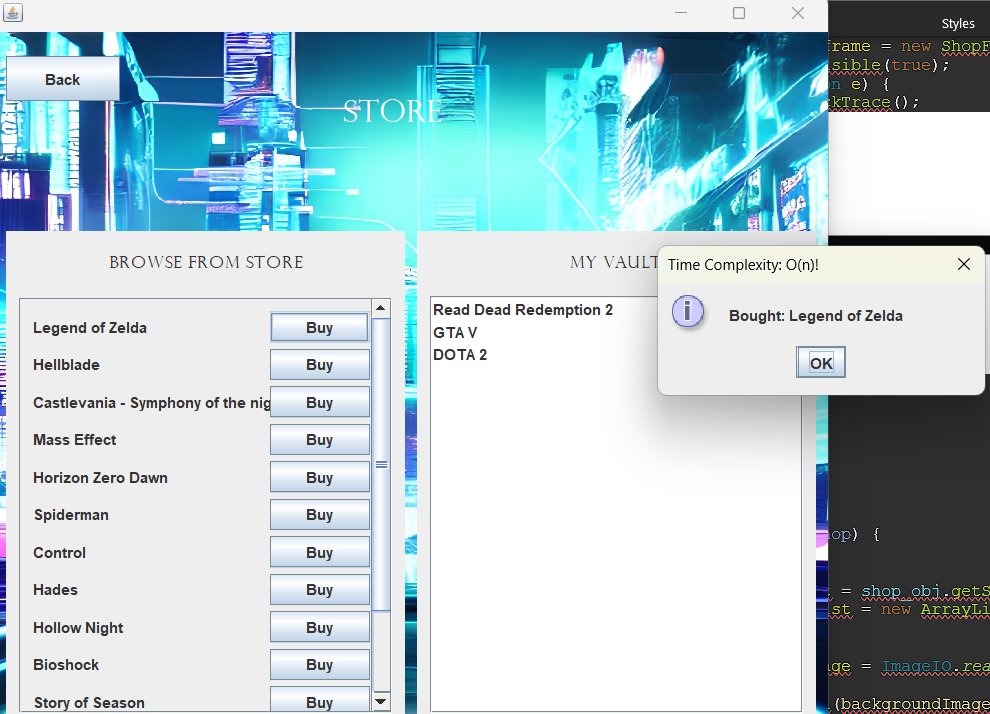
}

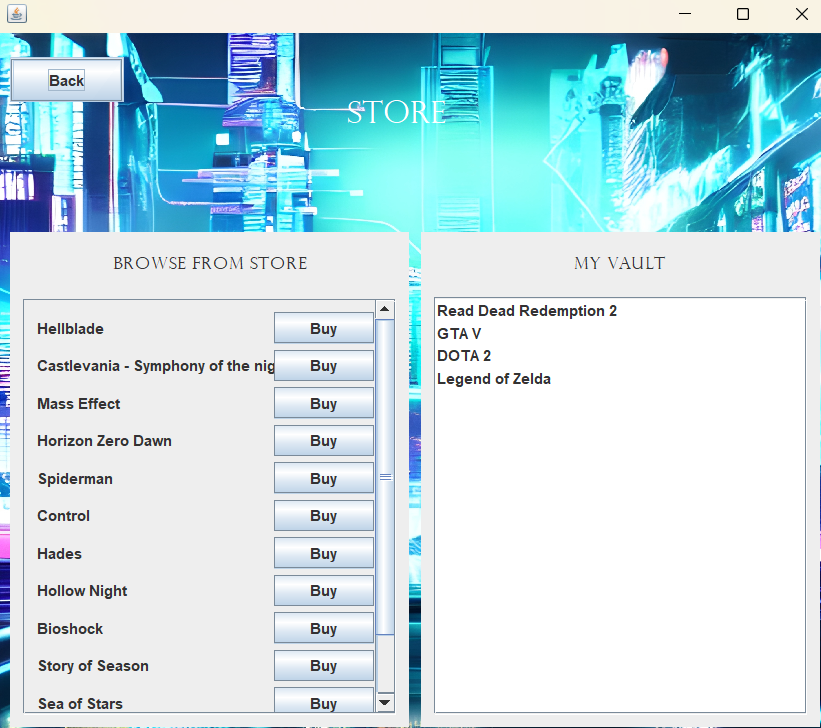
}

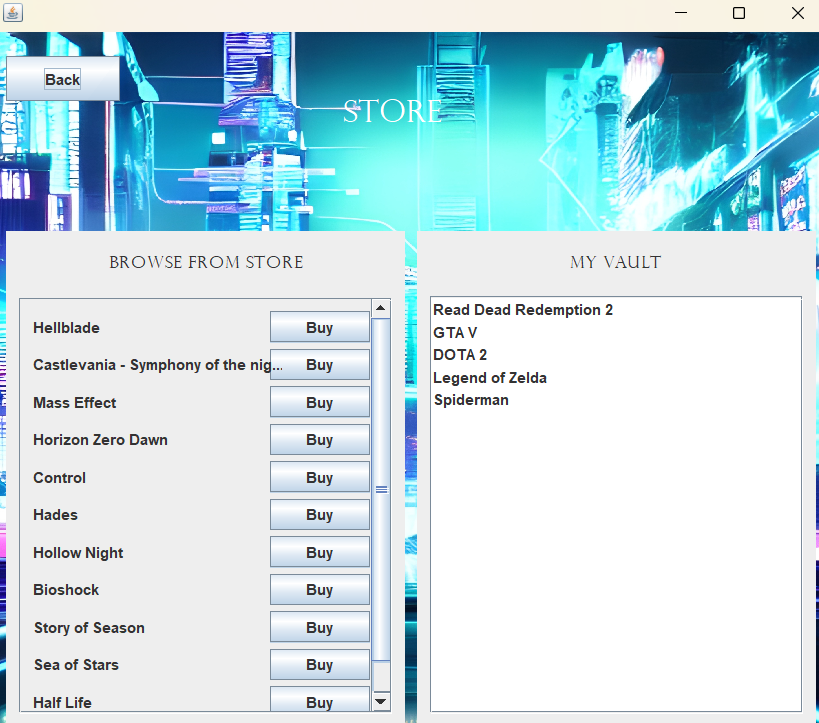
}

}









***1. AchievementsFrame Class:***

* The "AchievementsFrame" class is a graphical user interface that displays a user's achievements.
* It extends the "JFrame" class, which represents a window for a Java Swing application.

**2. Main Method:**

* The main method is used to create an instance of "AchievementsFrame" and make it visible.
* It takes a user as a parameter to display the achievements of that user.

**3. Constructor:**

* The constructor of the "AchievementsFrame" class initializes the frame.
* It sets the background image and creates various components like buttons, labels, and a scroll pane to display user achievements.

**4. MainPanel Class:**

* This is a nested class within "AchievementsFrame."
* It extends "JPanel" and is responsible for rendering the background image on the frame.
* It overrides the **paintComponent** method to draw the background image.

**5. Components and Layout:**

* The frame contains panels, labels, buttons, and a scroll pane to create the user interface.
* These components are arranged to display the user's achievements and milestones.

**6. Back Button:**

* There is a "Back" button that allows the user to go back to the previous screen.

**7. Scroll Pane for Achievements:**

* A scroll pane is used to display the list of achievements.
* The **updateAchievements** method updates the content in the scroll pane with the user's achievements.

**8. updateAchievements Method:**

* The **updateAchievements** method is responsible for populating the list of user achievements.
* It creates labels for each achievement and adds them to the scroll pane.

**9. Repainting and Refreshing:**

* After updating the achievements, the code uses **revalidate()** and **repaint()** to refresh the scroll pane and display the updated content.

**10. Handling Background Image:**

* The background image is loaded in the constructor using the **ImageIO** class. If the image fails to load, it falls back to using a regular JPanel.

In summary, the "AchievementsFrame" class provides a graphical user interface for displaying a user's achievements and milestones. It allows the user to see their accomplishments in a visually appealing way and includes a "Back" button to return to the previous screen.

public class AchievementsFrame extends JFrame {

private BufferedImage backgroundImage;

private static final long ***serialVersionUID*** = 1L;

private JPanel contentPane;

private User user\_obj; // Add a field to store the user object

private JScrollPane scrollPane;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.*invokeLater*(new Runnable() {

public void run() {

try {

AchievementsFrame frame = new AchievementsFrame(null);

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public AchievementsFrame(User user) {

this.user\_obj = user;

try {

backgroundImage = ImageIO.*read*(new File("src/img/achievements.jpg"));

} catch (IOException e) {

e.printStackTrace();

}

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setBounds(100, 100, 682, 603);

contentPane = new AchievementsMainPanel(); // Use the new AchievementsMainPanel instead of a regular JPanel

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 10, 668, 556);

contentPane.add(panel);

panel.setLayout(null);

panel.setOpaque(false);

JPanel panel\_1 = new JPanel();

panel\_1.setLayout(null);

panel\_1.setBounds(0, 0, 668, 140);

panel.add(panel\_1);

panel\_1.setOpaque(false);

JPanel panel\_2 = new JPanel();

panel\_2.setLayout(null);

panel\_2.setBounds(224, 24, 189, 68);

panel\_2.setOpaque(false);

panel\_1.add(panel\_2);

JLabel lblNewLabel = new JLabel("Milestones");

lblNewLabel.setHorizontalTextPosition(SwingConstants.***CENTER***);

lblNewLabel.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel.setForeground(Color.***BLACK***);

lblNewLabel.setFont(new Font("Perpetua Titling MT", Font.***PLAIN***, 25));

lblNewLabel.setBounds(10, 10, 169, 48);

panel\_2.add(lblNewLabel);

JButton btnNewButton\_3 = new JButton("Back");

btnNewButton\_3.setBounds(10, 10, 91, 36);

panel.add(btnNewButton\_3);

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose();

}

});

JPanel panel\_7\_1\_1 = new JPanel();

panel\_7\_1\_1.setLayout(null);

panel\_7\_1\_1.setBounds(0, 134, 668, 422);

panel.add(panel\_7\_1\_1);

panel\_7\_1\_1.setOpaque(false);

JPanel panel\_3 = new JPanel();

panel\_3.setBounds(10, 35, 648, 364);

panel\_7\_1\_1.add(panel\_3);

panel\_3.setLayout(null);

panel\_3.setOpaque(false);

scrollPane = new JScrollPane();

scrollPane.setBounds(10, 10, 628, 341);

panel\_3.add(scrollPane);

JLabel lblNewLabel\_2\_1\_1 = new JLabel("My Achievements");

lblNewLabel\_2\_1\_1.setForeground(Color.***BLACK***);

lblNewLabel\_2\_1\_1.setBounds(226, 10, 185, 29);

panel\_7\_1\_1.add(lblNewLabel\_2\_1\_1);

lblNewLabel\_2\_1\_1.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel\_2\_1\_1.setFont(new Font("Perpetua Titling MT", Font.***PLAIN***, 18));

updateAchievements();

}

public void updateAchievements() {

// Create a panel to hold the game levels

JPanel gameLevelsPanel = new JPanel();

gameLevelsPanel.setLayout(null);

// Get the user's game library

Stack<Game> userGameLibrary = this.user\_obj.getGamesLibrary();

int i = 1;

for (Game game : userGameLibrary) {

// Get the level of the game

int level = game.getLevel();

String title = game.getTitle();

// Create a label to display the level

JLabel levelLabel = new JLabel(i+". ("+title+") -> Level: " + level);

levelLabel.setBounds(10, 10 + (i-1) \* 30, 500, 25);

gameLevelsPanel.add(levelLabel);

i++;

}

gameLevelsPanel.setPreferredSize(new Dimension(628, (i-1) \* 30 + 30)); // Adjust the width and height

// Remove the old content from the scroll pane

scrollPane.getViewport().removeAll();

scrollPane.setRowHeaderView(gameLevelsPanel);

// Repaint the scroll pane to update the view

scrollPane.revalidate();

scrollPane.repaint();

}

class AchievementsMainPanel extends JPanel {

*@Override*

protected void paintComponent(Graphics g) {

super.paintComponent(g);

if (backgroundImage != null) {

g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);

}

}

}

}

A screenshot of a video game

Description automatically generated

**1. GamesFrame Class:**

* The "GamesFrame" class is a graphical user interface for managing and displaying a user's game vault.
* It extends the "JFrame" class, which represents a window for a Java Swing application.

**2. Main Method:**

* The main method is used to create an instance of "GamesFrame" and make it visible.
* It takes a user as a parameter to display and manage the user's game vault.

**3. Constructor:**

* The constructor of the "GamesFrame" class initializes the frame.
* It sets the background image and creates various components like buttons, text fields, and a list to display user games.

**4. MainPanel Class:**

* This is a nested class within "GamesFrame."
* It extends "JPanel" and is responsible for rendering the background image on the frame.
* It overrides the **paintComponent** method to draw the background image.

**5. Components and Layout:**

* The frame contains panels, buttons, labels, text fields, and a list to create the user interface.
* These components are arranged to add, delete, and display user games in a visually appealing way.

**6. Back Button:**

* There is a "Back" button that allows the user to go back to the previous screen.

**7. Tabbed Panes:**

* Tabbed panes are used to organize different functionalities: adding games, deleting games, and viewing games in the vault.

**8. Adding Games:**

* Users can add games by entering the game title in a text field and clicking the "Add to Vault" button.
* A new game is created and added to the user's game library. A pop-up message is displayed to confirm the addition.

**9. Deleting Games:**

* Users can delete games by searching for a game title and clicking the "Unvault" button.
* The game is removed from the user's game library, and a pop-up message is displayed to confirm the removal.

**10. Removing Previous Game:**

* Users can remove the most recently added game by clicking the "Unvault" button.
* The top game from the user's game library is removed, and a pop-up message confirms the removal.

**11. Displaying Games in the Vault:**

* A list is used to display the games currently in the user's game library.
* The "updateGamesInVault" method updates the list with the user's games.

In summary, the "GamesFrame" class provides a graphical user interface for managing and displaying a user's game vault. Users can add, delete, and view their games. It includes various features like pop-up messages for user interaction and offers an organized tabbed interface for different functionalities.

package soph\_part;

import java.awt.EventQueue;

import javax.imageio.ImageIO;

import javax.swing.DefaultListModel;

import javax.swing.JList;

import javax.swing.JOptionPane;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.GridBagLayout;

import java.awt.FlowLayout;

import javax.swing.SwingConstants;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.GridLayout;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.util.Iterator;

import java.util.List;

import java.util.Stack;

import java.awt.event.ActionEvent;

import javax.swing.JTabbedPane;

import javax.swing.JTextField;

import java.awt.Component;

public class GamesFrame extends JFrame {

private static final long ***serialVersionUID*** = 1L;

private JPanel contentPane;

private JTextField txt\_game\_title;

private JTextField txt\_game\_title\_delete\_search;

private JList<String> games\_list\_all\_obj;

private User current\_user;

private BufferedImage backgroundImage;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.*invokeLater*(new Runnable() {

public void run() {

try {

GamesFrame frame = new GamesFrame(null);

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public GamesFrame(User user) {

try {

backgroundImage = ImageIO.*read*(new File("src/img/games.jpg"));

} catch (IOException e) {

e.printStackTrace();

}

this.current\_user= user;//create user instance outside the button

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setBounds(100, 100, 682, 603);

contentPane = new MainPanel(); // Use the new MainPanel instead of a regular JPanel

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

setContentPane(contentPane);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 10, 668, 140);

contentPane.add(panel);

panel.setLayout(null);

panel.setOpaque(false);

JPanel panel\_2 = new JPanel();

panel\_2.setBounds(224, 24, 189, 68);

panel.add(panel\_2);

panel\_2.setLayout(null);

panel\_2.setOpaque(false);

JLabel lblNewLabel = new JLabel("Game Vault");

lblNewLabel.setFont(new Font("Perpetua Titling MT", Font.***PLAIN***, 25));

lblNewLabel.setForeground(Color.***WHITE***);

lblNewLabel.setBounds(10, 10, 169, 48);

// Center the label horizontally

lblNewLabel.setHorizontalAlignment(SwingConstants.***CENTER***);

panel\_2.add(lblNewLabel);

JButton btnNewButton\_3 = new JButton("Back");

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose();

}

});

btnNewButton\_3.setBounds(10, 10, 91, 36);

panel.add(btnNewButton\_3);

JPanel panel\_3 = new JPanel();

panel\_3.setBounds(0, 160, 668, 396);

contentPane.add(panel\_3);

panel\_3.setLayout(null);

panel\_3.setOpaque(false);

JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.***TOP***);

tabbedPane.setBounds(10, 10, 638, 376);

panel\_3.add(tabbedPane);

JPanel panel\_7 = new JPanel();

tabbedPane.addTab("Add Game", null, panel\_7, null);

tabbedPane.setEnabledAt(0, true);

panel\_7.setLayout(null);

JPanel panel\_4 = new JPanel();

panel\_4.setBounds(10, 52, 335, 119);

panel\_7.add(panel\_4);

panel\_4.setLayout(null);

JLabel lblNewLabel\_1 = new JLabel("Game Title");

lblNewLabel\_1.setHorizontalTextPosition(SwingConstants.***CENTER***);

lblNewLabel\_1.setHorizontalAlignment(SwingConstants.***LEFT***);

lblNewLabel\_1.setFont(new Font("Perpetua", Font.***PLAIN***, 15));

lblNewLabel\_1.setBounds(10, 10, 105, 29);

panel\_4.add(lblNewLabel\_1);

txt\_game\_title = new JTextField();

txt\_game\_title.setBounds(10, 46, 194, 29);

panel\_4.add(txt\_game\_title);

txt\_game\_title.setColumns(10);

JButton btn\_add\_game = new JButton("Add to Vault");

btn\_add\_game.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String gameTitle = txt\_game\_title.getText();

Game game\_obj = new Game(gameTitle);

JOptionPane.*showMessageDialog*(null, "Added: "+gameTitle, "Time Complexity: O(1)", JOptionPane.***INFORMATION\_MESSAGE***);

current\_user.addGameToLibrary(game\_obj);

System.***out***.print(current\_user.getGamesLibrary());

updateGamesInVault(current\_user.getGamesLibrary());

}

});

btn\_add\_game.setBounds(10, 85, 123, 21);

panel\_4.add(btn\_add\_game);

JPanel panel\_5 = new JPanel();

panel\_5.setAlignmentY(Component.***TOP\_ALIGNMENT***);

panel\_5.setAlignmentX(Component.***LEFT\_ALIGNMENT***);

panel\_5.setBounds(222, 10, 170, 32);

panel\_7.add(panel\_5);

panel\_5.setLayout(null);

JLabel lblNewLabel\_2 = new JLabel("New Game Entry");

lblNewLabel\_2.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel\_2.setFont(new Font("Perpetua Titling MT", Font.***PLAIN***, 14));

lblNewLabel\_2.setBounds(0, 0, 170, 29);

panel\_5.add(lblNewLabel\_2);

JPanel panel\_7\_1 = new JPanel();

panel\_7\_1.setLayout(null);

tabbedPane.addTab("Delete Game", null, panel\_7\_1, null);

JPanel panel\_4\_1 = new JPanel();

panel\_4\_1.setLayout(null);

panel\_4\_1.setBounds(10, 52, 297, 119);

panel\_7\_1.add(panel\_4\_1);

JLabel lblNewLabel\_1\_1 = new JLabel("Search Game Title");

lblNewLabel\_1\_1.setHorizontalTextPosition(SwingConstants.***CENTER***);

lblNewLabel\_1\_1.setHorizontalAlignment(SwingConstants.***LEFT***);

lblNewLabel\_1\_1.setFont(new Font("Perpetua", Font.***PLAIN***, 15));

lblNewLabel\_1\_1.setBounds(10, 10, 111, 29);

panel\_4\_1.add(lblNewLabel\_1\_1);

txt\_game\_title\_delete\_search = new JTextField();

txt\_game\_title\_delete\_search.setColumns(10);

txt\_game\_title\_delete\_search.setBounds(10, 46, 194, 29);

panel\_4\_1.add(txt\_game\_title\_delete\_search);

JButton btn\_delete\_game\_search = new JButton("Unvault");

btn\_delete\_game\_search.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String game\_title = txt\_game\_title\_delete\_search.getText();

Stack<Game> games\_list = current\_user.getGamesLibrary();

Iterator<Game> iterator = games\_list.iterator();

while (iterator.hasNext()) {

Game game = iterator.next();

if (game.getTitle().equals(game\_title)) {

JOptionPane.*showMessageDialog*(null, "Removed: "+game.getTitle().toString(), "Time Complexity: O(n)!", JOptionPane.***WARNING\_MESSAGE***);

iterator.remove();

}

}

updateGamesInVault(games\_list);

}

});

btn\_delete\_game\_search.setBounds(10, 85, 119, 24);

panel\_4\_1.add(btn\_delete\_game\_search);

JPanel panel\_5\_1 = new JPanel();

panel\_5\_1.setLayout(null);

panel\_5\_1.setAlignmentY(0.0f);

panel\_5\_1.setAlignmentX(0.0f);

panel\_5\_1.setBounds(222, 10, 170, 32);

panel\_7\_1.add(panel\_5\_1);

JLabel lblNewLabel\_2\_1 = new JLabel("Delete Game Entry");

lblNewLabel\_2\_1.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel\_2\_1.setFont(new Font("Perpetua Titling MT", Font.***PLAIN***, 14));

lblNewLabel\_2\_1.setBounds(0, 0, 170, 29);

panel\_5\_1.add(lblNewLabel\_2\_1);

JPanel panel\_4\_1\_1 = new JPanel();

panel\_4\_1\_1.setLayout(null);

panel\_4\_1\_1.setBounds(326, 52, 297, 119);

panel\_7\_1.add(panel\_4\_1\_1);

JLabel lblNewLabel\_1\_1\_1 = new JLabel("Remove Previous Game");

lblNewLabel\_1\_1\_1.setHorizontalTextPosition(SwingConstants.***CENTER***);

lblNewLabel\_1\_1\_1.setHorizontalAlignment(SwingConstants.***LEFT***);

lblNewLabel\_1\_1\_1.setFont(new Font("Perpetua", Font.***PLAIN***, 15));

lblNewLabel\_1\_1\_1.setBounds(10, 10, 144, 29);

panel\_4\_1\_1.add(lblNewLabel\_1\_1\_1);

JButton btn\_delete\_game = new JButton("Unvault");

btn\_delete\_game.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Stack<Game> game\_list = current\_user.getGamesLibrary();

if(!game\_list.isEmpty()) {

Game removedGame = game\_list.pop();

JOptionPane.*showMessageDialog*(null, "Removed: "+removedGame.getTitle().toString(), "Time Complexity: O(1)!", JOptionPane.***WARNING\_MESSAGE***);

updateGamesInVault(game\_list);

}

}

});

btn\_delete\_game.setBounds(10, 46, 194, 29);

panel\_4\_1\_1.add(btn\_delete\_game);

JPanel panel\_7\_1\_1 = new JPanel();

panel\_7\_1\_1.setLayout(null);

tabbedPane.addTab("Games in the Vault", null, panel\_7\_1\_1, null);

JPanel panel\_5\_1\_1 = new JPanel();

panel\_5\_1\_1.setLayout(null);

panel\_5\_1\_1.setAlignmentY(0.0f);

panel\_5\_1\_1.setAlignmentX(0.0f);

panel\_5\_1\_1.setBounds(222, 10, 170, 32);

panel\_7\_1\_1.add(panel\_5\_1\_1);

JLabel lblNewLabel\_2\_1\_1 = new JLabel("Explore Games");

lblNewLabel\_2\_1\_1.setHorizontalAlignment(SwingConstants.***CENTER***);

lblNewLabel\_2\_1\_1.setFont(new Font("Perpetua Titling MT", Font.***PLAIN***, 14));

lblNewLabel\_2\_1\_1.setBounds(0, 0, 170, 29);

panel\_5\_1\_1.add(lblNewLabel\_2\_1\_1);

games\_list\_all\_obj = new JList<>(); //list of games by the user

JScrollPane scrollPane = new JScrollPane(games\_list\_all\_obj);

scrollPane.setBounds(10, 52, 613, 287);

panel\_7\_1\_1.add(scrollPane);

JPanel panel\_1 = new JPanel();

int centerX = (panel.getWidth() - panel\_1.getWidth()) / 2;

int centerY = (panel.getHeight() - panel\_1.getHeight()) / 2;

panel\_1.setBounds(centerX, centerY, 200, 100);

}

public void updateGamesInVault(Stack<Game> games) {

DefaultListModel<String> gameListModel = new DefaultListModel<>();

for (Game game : games) {

gameListModel.addElement(game.getTitle());

}

games\_list\_all\_obj.setModel(gameListModel);

}

class MainPanel extends JPanel {

*@Override*

protected void paintComponent(Graphics g) {

super.paintComponent(g);

if (backgroundImage != null) {

g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);

}

}

}

}

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated