CS 319 - Object-Oriented Software Engineering System Design Report Iteration 1

CS CRUSH

<u>Group 1-A</u>

Ahmet Akif Uğurtan

Taner Düzceer

Berk Ataç

Eren Aytüre

1.Introduction

1.1 Purpose of the System

There are a lot of puzzle video games out there with different kind of objectives. CSCrush is a 2-D match-three puzzle type video game we decided to develop as desktop application. The user can play this game by using mouse. In our game, user try to reach a specific points which is given by the game according to the level. The user should change the places of the icon of different type of books. The user can change the place of books if the books are neighbor each other vertically or horizontally. The user tries to catch at least three the same order of same type of books vertically or horizontally. When the user can order like this, the user get points and the ordered books are disappeared. Therefore, this game tests the user's problem solving skills. When the user pass a level of this game, another level is opened. This level is more difficult than the previous one. Moreover, there are some power-ups which are help the user. The user can use them in difficult circumstances.

The game we were influenced by is Candy Crush.https://king.com/tr/game/candycrush CSCrush will not be an exact replica of Candy Crush. In CSCrush classic matching-three game rules will apply. Player will be presented with a level full of objects and try to match as many as possible, using various power-ups, in order to earn score then advance to the next levels and there will be finite amount of moves for the player. Players who complete levels will be on high score board.

1.2 Design Goals

The design goals are important to meet the qualification of the application. So, we paid our attention to functional and nonfunctional components of the system. Important design goals are listed below

Performance Criteria:

Response Time and Throughput

The game is designed to satisfy response time. This game also requires instant response to maintain a good play. Namely, when the user click on an icon and another icon, the game should swap them fast. The application is going to be responsive and it will work with high performance.

Memory

This application does not require big amount of memory. The memory is used for only functional and nonfunctional components of the system.

Dependability Criteria:

Reliability

The applicants hasn't got bugs and inconsistency. For example, the user can't swap the icons which are not neighbor.

Cost Criteria:

Development Cost

This app is not a commercial product. The system which the application is developed is not expensive. The external components of the system like images, icons, vs. obtained from the existing free resources. CSCrush is not a commercial product. This is because, this application can be installed and used freely.

Maintenance Criteria:

Extensibility

Adding some new features is important for the future of the game. Moreover, the users attention can be gained by adding new features. These features can be new power ups, new type of book icons vs. On the other hand, the Appleton can be developed according to the user's suggestions.

Modifiability

It is possible to modify our game according to system requirements. Our application system requirements are low but, we can modify it according to any system.

Portability

Our application is going to be developed by using java. Java works on portable Java Virtual Machine(JVM). This is because, this platform can work on different machines. Therefore, our application is portable.

Readability

The designation of application is simple. In that sense, the code of the application can be understandable. To improve the understandability, we will add extra comments. Moreover, Object oriented software is very effective. We can design any objects on the way of requirements.

End User Criteria:

Usability

The simplicity of the usage of the game is important design goal because the easiness of usage provide us user friendliness and the user’s attraction. The simplicity of the game is related to user interface. In our game, the user can play it easily. Swapping the icons can be done by clicking on the mouse. Power ups will be on the screen. Namely, the user can reach every components of the game easily. Moreover, the simplicity of the user interface is related to the simplicity of the design of the game.

Design Goal Trade Offs:

Response Time and Throughput vs. Reusability

Response time and throughput is important for our application because the user should take reaction when he/she click on something which is related to game. However, we don’t think that our game will not integrate any other applications

Space vs. Speed

Our game is interactive and the performances is important. Therefore, Providing the performance for our game, we need to use memory. The memory is used for book objects, power ups and the final status of the game, the levels which are the users in.

Definitions, acronyms and abbreviations

Cross-platform

It means that a program is able to run in different platforms such as Windows, Linux, MAC OS properly in a same way. [1]

JRE

Java Runtime Environment in short JRE is a set of software tools for Java applications and Java programming. [2]

FPS

Frames Per Second in short FPS is the number of different frames in one second. Animations are usually 60 FPS since it seems better and nicer to human eye. [3]

References

https://king.com/tr/game/candycrush

https://en.oxforddictionaries.com/definition/cross-platform

https://www.techopedia.com/definition/5442/java-runtime-environment-jre

https://www.urbandictionary.com/define.php?term=fps

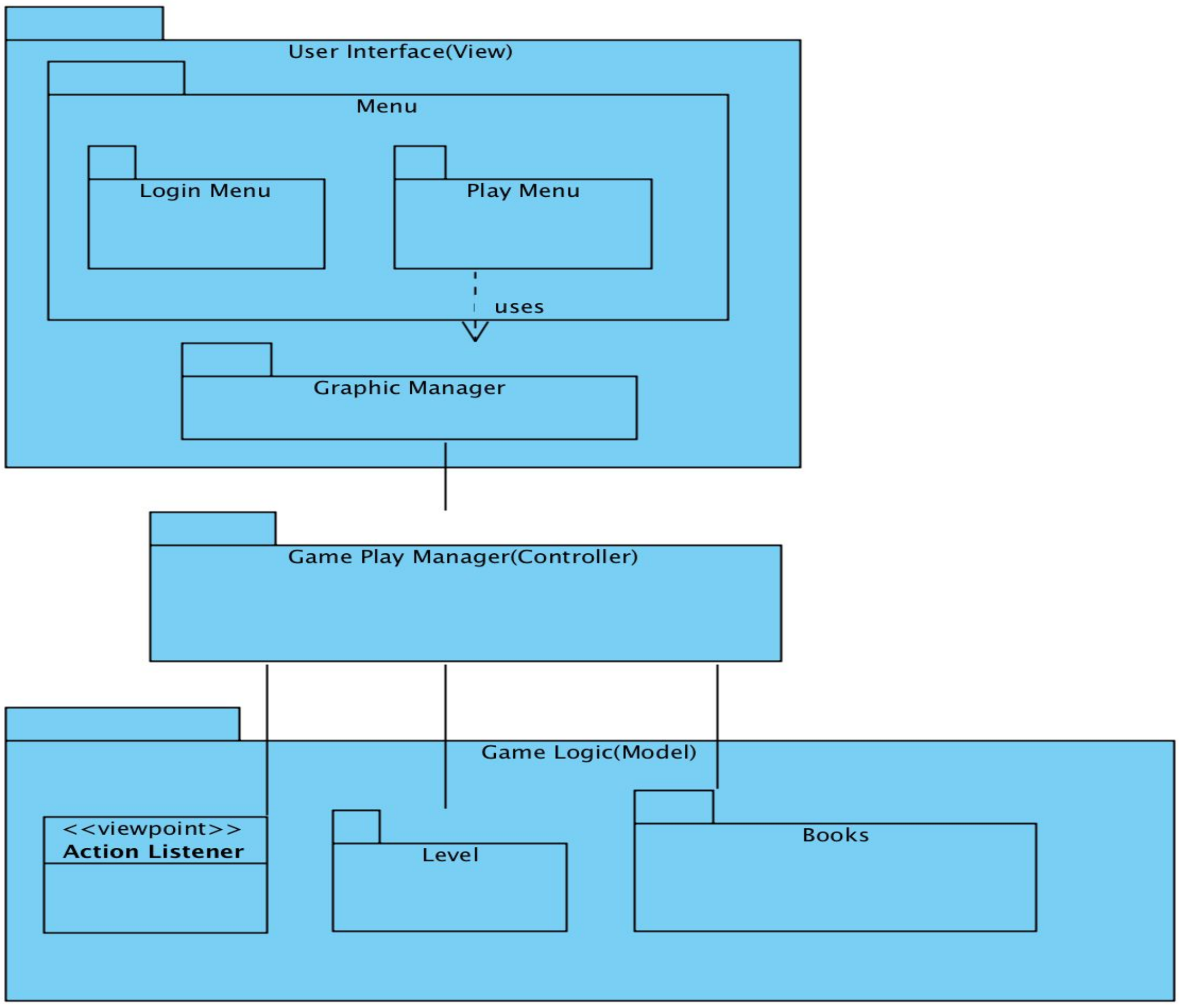
2. Software Architecture

2.1 Overview

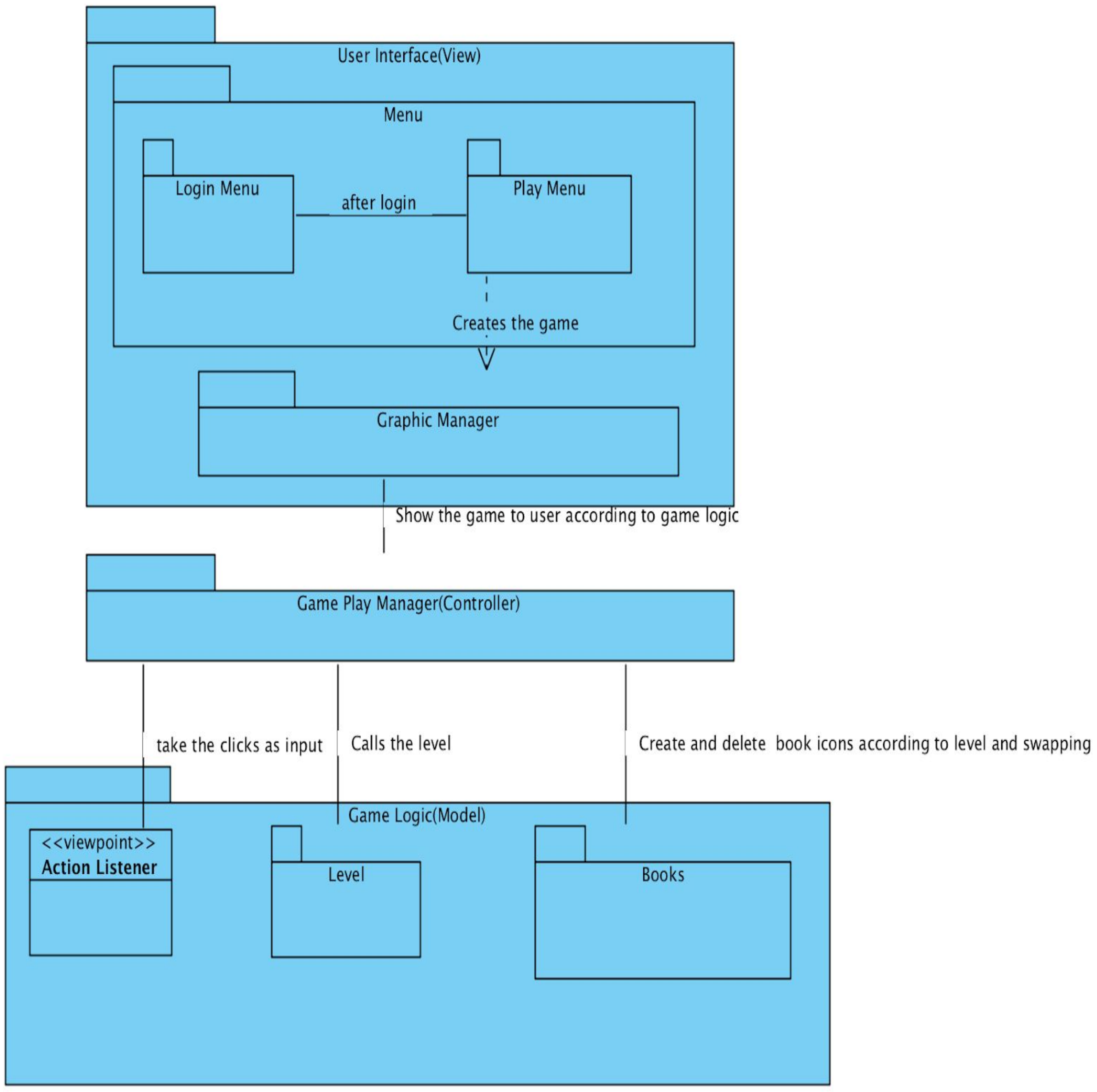
This section will show the software architecture of the system. The Main System will decompose into subsystems in a Three layered architectural style. Thus, the system will be more controllable.

2.2 Subsystem Decomposition

By dividing Main system into several subsystems, the program will be more modifiable and extendible. There are three subsystems named User Interface, Game Logic and Storage. This flow shows the logic from user interface to game logic to storage. Subsystems have their entities with similar funcitonalities and arrows shows the hierarchy in subsystems. Thanks to subsystems and workflow, the program will meet the non-functional requirements. If there needs any modification to improve the program efficiently, this architectural path will allow us to modify program without contradictions.



UI and Gameplay Layer interactions(Basic)



UI and Gameplay Layer interactions(Detailed)

2.3 Architectural Styles

2.3.1 User Interface (View)

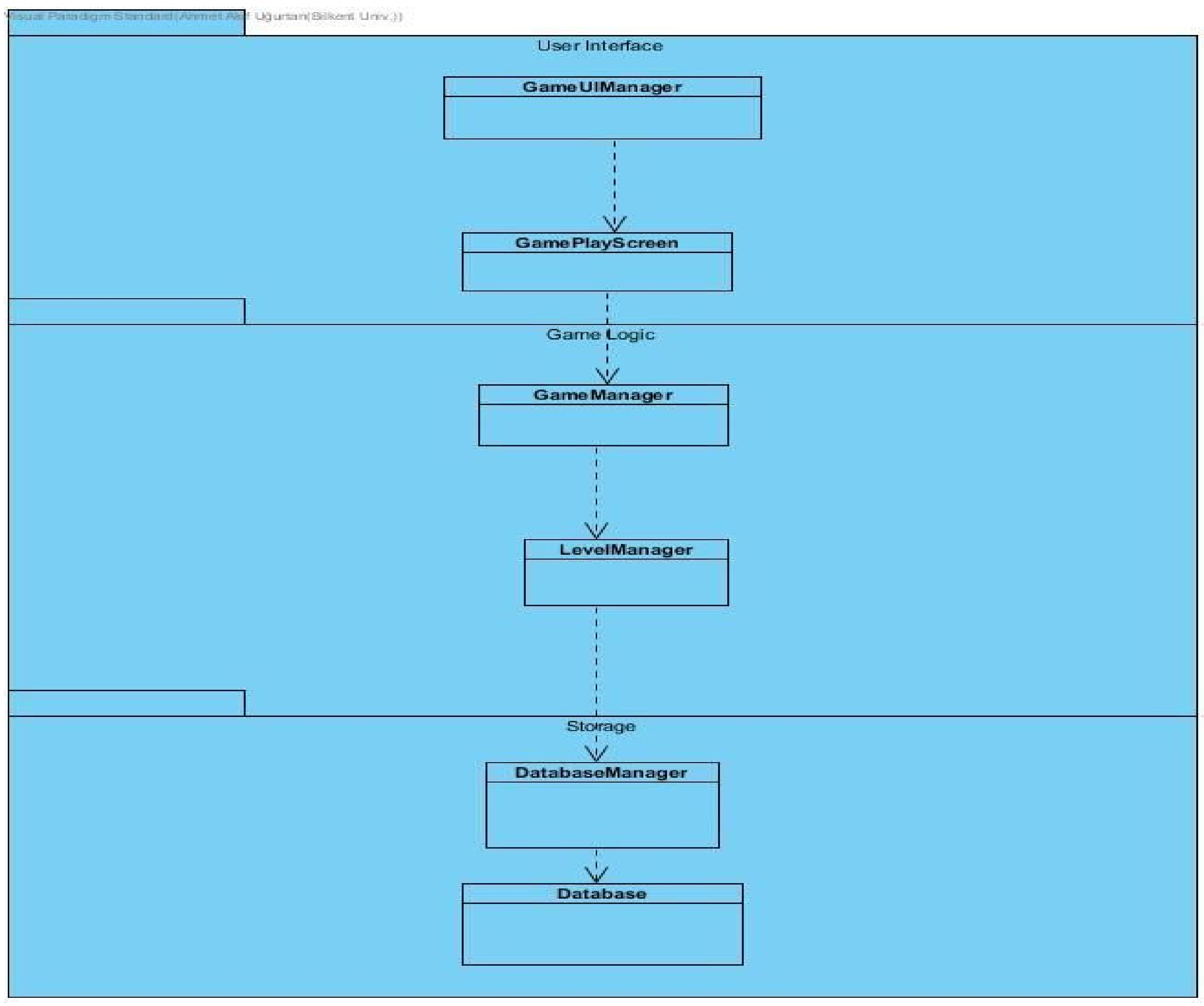
The CsCrush game has a user interface component. This component is used to show the game to user. In the login menu, the user can see login, sign up and buttons. After login, The user sees the levels. Shortly, after seeing these menus, the game panels and mouse clicking movements can be seen by the user.

2.3.2 Game Play Manager (Controller)

This component arranges the interaction between the user interface and Game logic. According to the game logic, the user interface shows the game to the user.

2.3.3 Game Logic(Model)

Game logic component determines the last situation of the game by user inputs. For example, the user wants to reach Login menu or play menu, this components executes the necessary calculations. Moreover, while playing the game, this components gives a decision about which books are disappeared or which books come in. Shortly, this component models the game on the way of user' inputs.



2.4 Hardware/Software Mapping

Since the game will be written in JAVA, we will use JRE (Java Runtime Environment). Thanks to JRE, it will be able to run at any device which has JRE. Game will run on RAM and one mouse is enough to play active game but a keyboard is necessary to write username and password while login or signup. In order to check user properties or to update new high score or to display high score table, the program will communicate with database which will run on another computer serving as a JAVA server.

2.5 Persistent Data Management

Users have to login every time to play game so that after sign up, user properties will be stored in a database. While login, the program will communicate with the database in order to check user credentials and to get user properties such as high score of user for each level.

User is able to display high scores by opening the high score interface via the button in main menu, the system must get best high scores and rank of user from database since user is challenging with other users and there must be synchronization.

After each breaking the record of high score of a level, the system will update the high score of this level in database for current user.

2.6 Access Control and Security

In order to play game, users have to have game .jar file and also internet connection. Nowadays, connecting the internet is not a big issue so we thought this will not be a problem and thanks to internet users will able to challenge each other by trying to get more points than others or themselves. In order to separate users from each other, each user has to have username and password. Also an email address is necessary for verifying user to prevent bot users. The system will not share any user credentials with public to ensure security and privacy.

2.7 Global Software Control

Event-driven Control

In our program, anything will be updated after clicking buttons or swiping images in game screen. Those button or mouse events will be controlled thanks to listeners in JAVA. GameUIManager will get responses from other UI classes and will decide what to do and GameManager will get changes from GameUIManager and it will update background data.

2.8 Boundary Conditions

Start-up

JAVA 8 or higher must be installed at device since game runs in JAVA. User has to have internet connection and game .jar file to play game.

Shutdown

User can terminate game by clicking the “Exit” button in main menu.

Exception Handling

User will be notified in case of any crashes such as not loading an image, not connecting the database, not having an internet connection.

3 Subsystem Services

3.1 Design Pattern

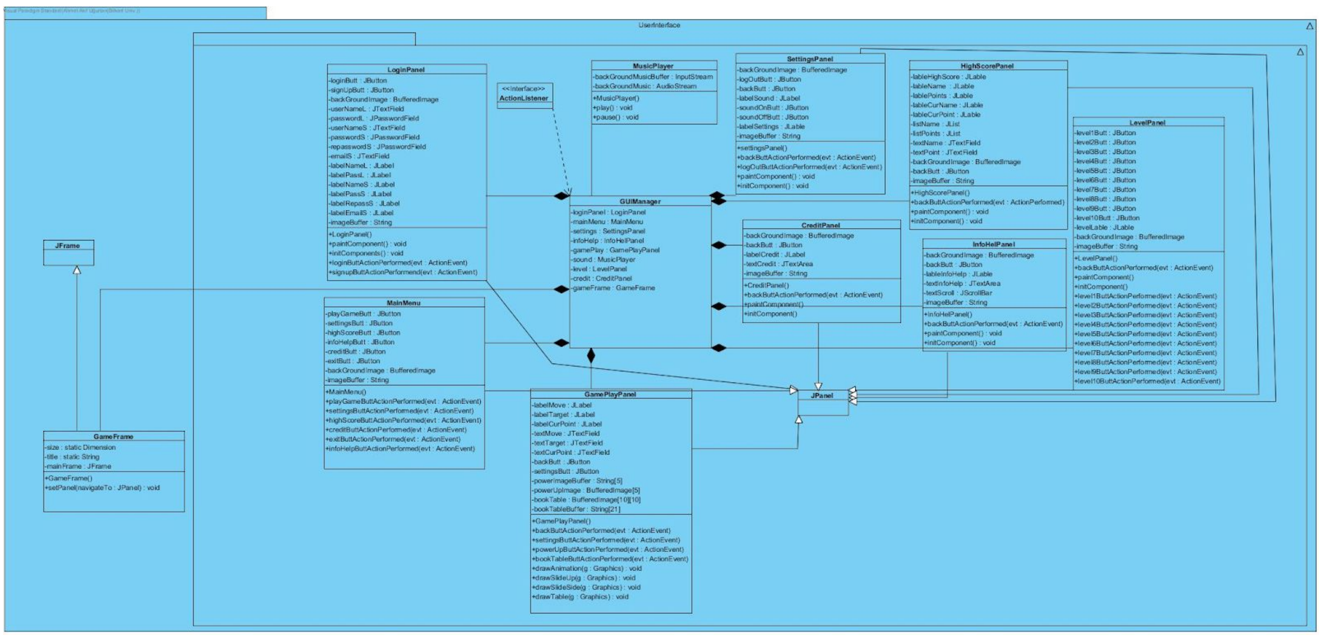
Behavioral Patterns

Façade

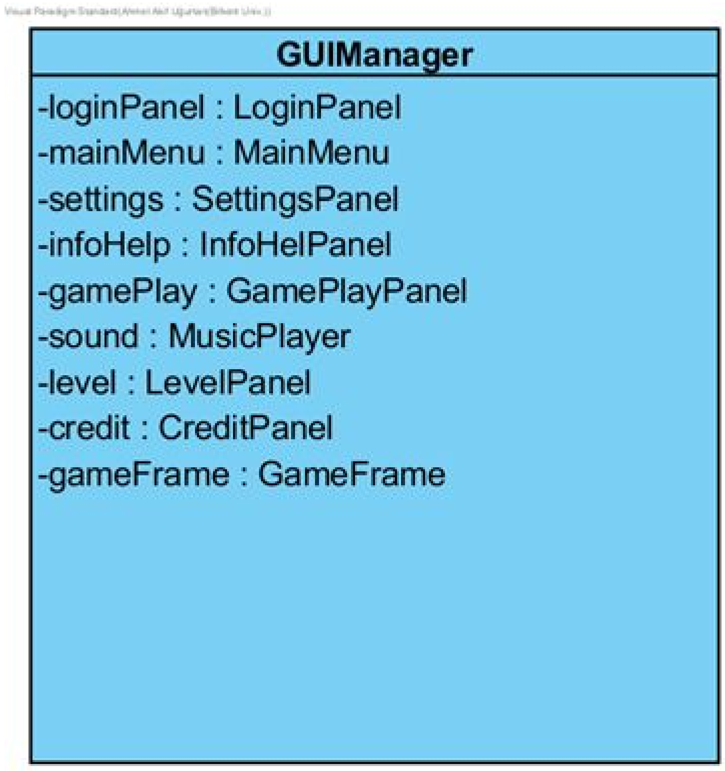
We will use Façade design pattern to create an interface for GameplayPanel class. GameManager is the Façade class which calls all other update methods. By doing this, we delegated the calls from GameplayPanel to game logic classes. This provides us more flexible working areas.

3.2 Designs

3.2.1 User Interface Subsystem



GUIManager Class



Attributes

private LoginPanel loginPanel : Instance of LoginPanel class which is created and managed by GUIManager class to set main panel of gameFrame as this class if user navigates to login page.

private MainMenu mainMenu : Instance of MainMenu class which is created and managed by GUIManager class to set main panel of gameFrame as this class if user navigates to main menu page.

private Settings settings : Instance of Settings class which is created and managed by GUIManager class to set main panel of gameFrame as this class if user navigates to settings page.

private InfoHelPanel infoHelp : Instance of InfoHelPanel class which is created and managed by GUIManager class to set main panel of gameFrame as this class if user navigates to information and help page.

private GamePlayPanel gamePlay : Instance of GamePlayPanel class which is created and managed by GUIManager class to set main panel of gameFrame as this class if user navigates to game playing page.

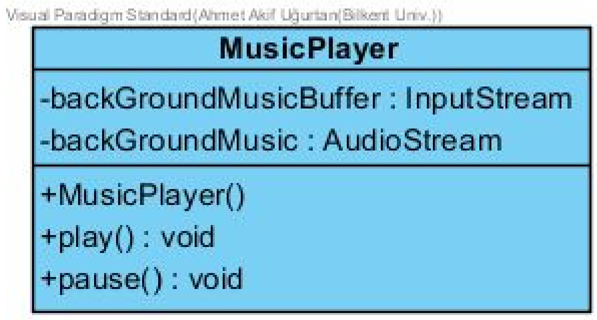
private MusicPlayer sound : Instance of MusicPlayer class which is created and managed by GUIManager class to play music on background in every page.

private LevelPanel level : Instance of LevelPanel class which is created and managed by GUIManager class to set main panel of gameFrame as this class if user navigates to level page.

private CreditPanel credit : Instance of CreditPanel class which is created and managed by GUIManager class to set main panel of gameFrame as this class if user navigates to credits page.

private GameFrame gameFrame : Instance of GameFrame class which is created and managed by GUIManager class to display panels during navigation between them.

MusicPlayer Class


Visual Paradigm Standard(Ahmet Akif Uğurtan(Bilkent Univ.))
MusicPlayer
-backGroundMusicBuffer : InputStream
-backGroundMusic : AudioStream
+MusicPlayer()
+play() : void
+pause() : void

Attributes

private InputStream backGroundMusicBuffer : This attribute saves an InputStream in order to be ready to play music.

private AudioStream backFroundMusic : After input stream is received this instance uses AudioStream class to start playing music.

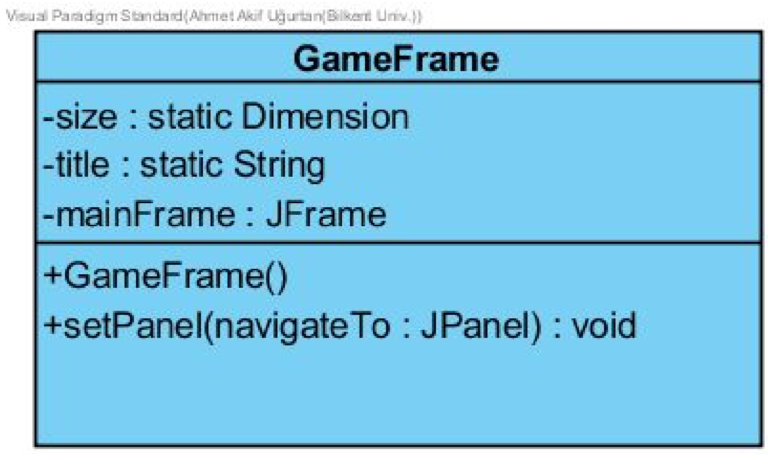
Methods

public MusicPlayer() : Constructor for MusicPlayer class to initialize an instance of MusicPlayer.

public void play() : A method to start playing music.

public void pause() : A method to pause music.

GameFrame Class


Visual Paradigm Standard(Ahmet Akif Uğurtan(Bilkent Univ.))
GameFrame
-size : static Dimension
-title : static String
-mainFrame : JFrame
+GameFrame()
+setPanel(navigateTo : JPanel) : void

Attributes

private JFrame mainFrame : Only Frame object in whole application to display panels.

private static Dimension size : A variable to save default frame size.

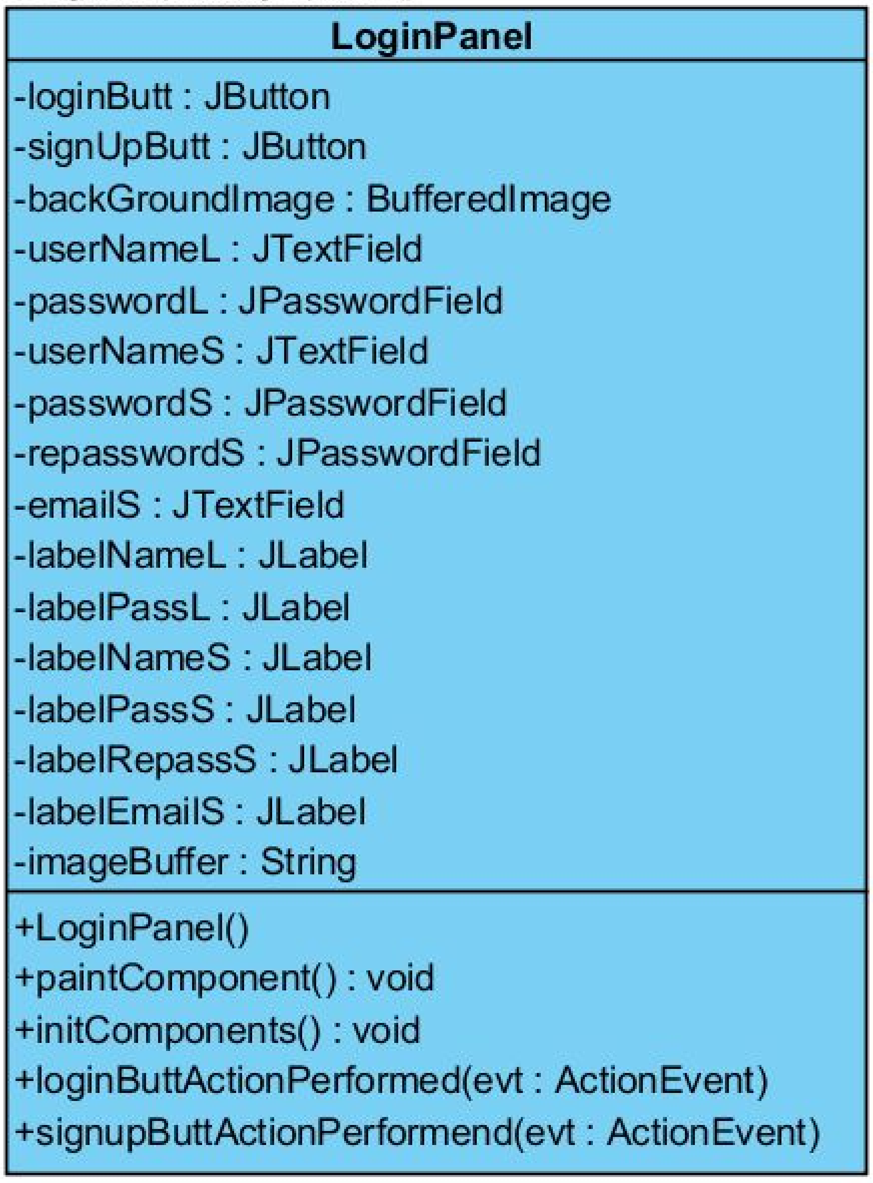
private static String title : Name of the game to be displayed on top of the frame.

Methods

public GameFrame() : Constructor for GameFrame class to initialize instance and create a JFrame.

public void setPanel(JPanel navigateTo) : This method changes the current panel object with navigateTo object so that the content in mainFrame changes and user navigates to desired page.

LoginPanel Class



Attributes

private JButton loginButt : Instance of JButton class to listen whether user wants to login or not.

private JButton signUpButt : Instance of JButton class to listen whether user wants to signup or not.

private JTextField userNameL : Instance of JTextField to get username from user when he/she clicks the loginButt button.

private JPasswordField passwordL : Instance of JPasswordField to get password from user when he/she clicks the loginButt button.

private JTextField userNameS : Instance of JTextField to get username from user when he/she clicks the singUpButt button.

private JPasswordField passwordS : Instance of JPasswordField to get password from user when he/she clicks the signUpButt button.

private JPasswordField repasswordS : Instance of JPasswordField to get password again to check with passwordS from user when he/she clicks the signUpButt button.

private JTextField emailS : Instance of JTextField to get email from user when he/she clicks the singUpButt button.

private JLabel labelNameL : Instance of JLabel to display “Username” label before userNameL text field to make user comprehend what he/she should enter in the text field.

private JLabel labelPassL : Instance of JLabel to display “Password” label before passwordL text field to make user comprehend what he/she should enter in the text field.

private JLabel labelNameS : Instance of JLabel to display “Username” label before userNameS text field to make user comprehend what he/she should enter in the text field.

private JLabel labelPassS : Instance of JLabel to display “Password” label before passwordS text field to make user comprehend what he/she should enter in the text field.

private JLabel labelRepassS : Instance of JLabel to display “Repassword” label before repasswordS text field to make user comprehend what he/she should enter in the text field.

private JLabel labelEmailS : Instance of JLabel to display “Email” label before emailS text field to make user comprehend what he/she should enter in the text field.

private BufferedImage backgroundImage : An instance to get and save background image then display in the screen.

private String imageBuffer : Direction of background image file.

Methods

public LoginPanel() : Constructor for LoginPanel class to initialize attributes.

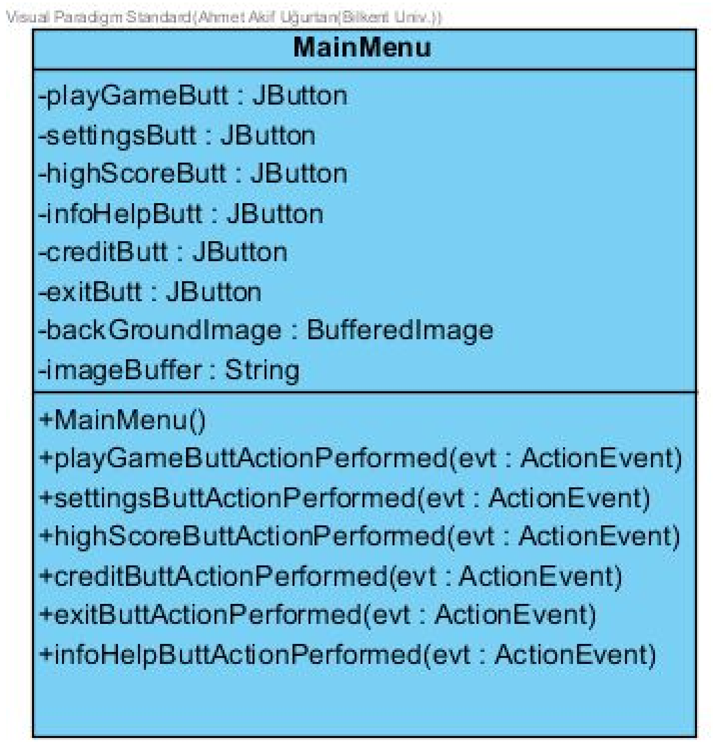
public void paintComponent() : Overriding javax.swing.JPanel method to paint objects over and over again.

public void initComponents() : This method initialize attributes by being called from constructor.

public void loginButtActionPerformed(ActionEvent evt) : Actionlistener method for loginButt button in order to post a request to server and login.

public void signupButtActionPerformed(ActionEvent evt) : Actionlistener method for signUpButt button in order to post a request to server and sign up.

MainMenu Class


Visual Paradigm Standard(Ahmet Akif Uğurtan(Bilkent Univ.))
MainMenu
-playGameButt : JButton
-settingsButt : JButton
-highScoreButt : JButton
-infoHelpButt : JButton
-creditButt : JButton
-exitButt : JButton
-backGroundImage : BufferedImage
-imageBuffer : String
+MainMenu()
+playGameButtActionPerformed(evt : ActionEvent)
+settingsButtActionPerformed(evt : ActionEvent)
+highScoreButtActionPerformed(evt : ActionEvent)
+creditButtActionPerformed(evt : ActionEvent)
+exitButtActionPerformed(evt : ActionEvent)
+infoHelpButtActionPerformed(evt : ActionEvent)

Attributes

private JButton playGameButt : A JButton instance to navigate Level page then playGame screen.

private JButton settingsButt : A JButton instance to navigate Settings page.

private JButton highScoreButt : A JButton instance to navigate High Score page.

private JButton infoHelpButt : A JButton instance to navigate Information and Help page.

private JButton creditButt : A JButton instance to navigate Credits page.

private JButton exitButt : A JButton instance to Exit and terminate game.

private BufferedImage backGroundImage : An instance o BufferedImage to get image and later to display background image.

private String imageBuffer : The direction of image file for background.

Methods

public MainMenu() : Constructor for MainMenu class to initialize objects.

public void playGameButtActionPerformed(ActionEvent evt) : Button listener for playGameButt.

public void settingsButtActionPerformed(ActionEvent evt) : Button listener for settingsButt.

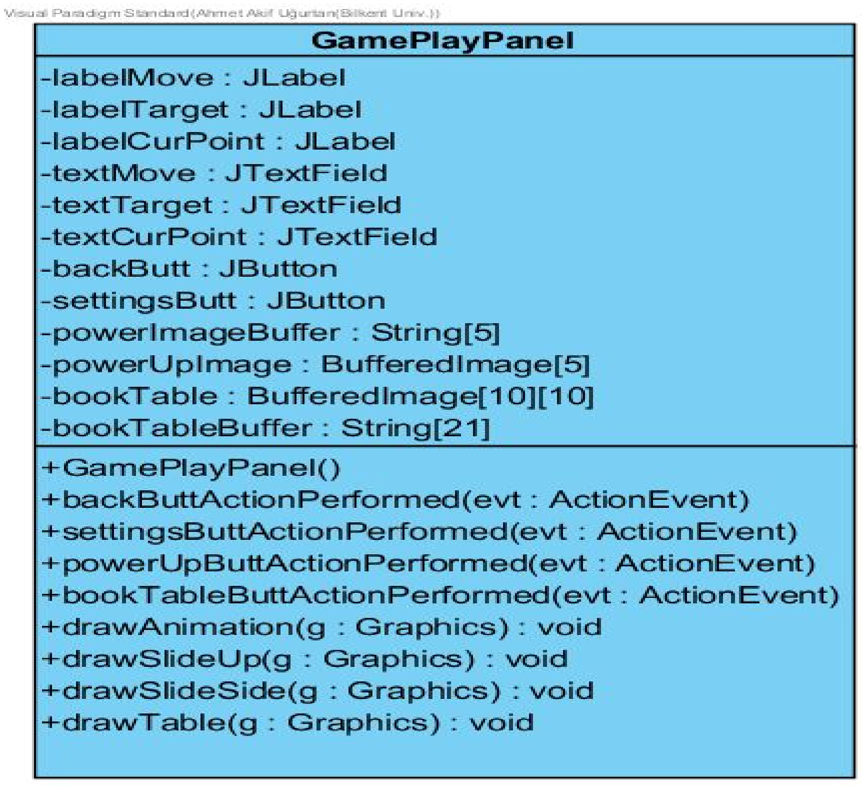
public void highScoreButtActionPerformed(ActionEvent evt) : Button listener for highScoreButt.

public void infoHelpButtActionPerformed(ActionEvent evt) : Button listener for infoHelpButt.

public void creditButtActionPerformed(ActionEvent evt) : Button listener for creditButt.

public void exitButtActionPerformed(ActionEvent evt) : Button listener for exitButt.

GamePlayPanel Class



```
GamePlayPanel
-labelMove : JLabel
-labelTarget : JLabel
-labelCurPoint : JLabel
-textMove : JTextField
-textTarget : JTextField
-textCurPoint : JTextField
-backButt : JButton
-settingsButt : JButton
-powerImageBuffer : String[5]
-powerUpImage : BufferedImage[5]
-bookTable : BufferedImage[10][10]
-bookTableBuffer : String[21]
+GamePlayPanel()
+backButtActionPerformed(evt : ActionEvent)
+settingsButtActionPerformed(evt : ActionEvent)
+powerUpButtActionPerformed(evt : ActionEvent)
+bookTableButtActionPerformed(evt : ActionEvent)
+drawAnimation(g : Graphics) : void
+drawSlideUp(g : Graphics) : void
+drawSlideSide(g : Graphics) : void
+drawTable(g : Graphics) : void
```

Attributes

private JLabel labelMove : Instance of JLabel to display “Remained Move” label before textMove text field to make user comprehend what that field means.

private JLabel labelTarget : Instance of JLabel to display “Target Point” label before textTarget text field to make user comprehend what that field means.

private JLabel labelCurPoint : Instance of JLabel to display “Current Point” label before textCurPoint text field to make user comprehend what that field means.

private JTextField textMove : Instance of JTextFied to display number of remained movements.

private JTextField textTarget : Instance of JTextFied to display target point.

private JTextField textCurPoint : Instance of JTextFied to display current achieved point.

private JButton backButt : A JButton instance to navigate back to the Level page.

private JButton settingsButt : A JButton instance to navigate Settings page.

private String powerImageBuffer[5] : Directions of images of power up objects.

private BufferedImage powerUpImage[5] : An instance array to get and save images of powerup objets.

private BufferedImage bookTable[10][10] : An instance array to get and save images of CSCrush objets. This array is the main game play table since all CSCrush objects and sliding, crushing animations will be represented here.

private String bookTableBuffer[21] : Directions of images of all CSCrush objects such as books, horizontal lined books, vertical lined books etc.

Methods

public GamePlayPanel() : Constructor for GamePlayPanel Class to initialize attributes.

public void backButtActionPerformed(ActionEvent evt) : Action listener for backButt.

public void settingsButtActionPerformed(ActionEvent evt) : Action listener for settingsButt.

public void powerUpButtActionPerformed(ActionEvent evt) : Action listener for powerUp images.

public void bookTableButtActionPerformed(ActionEvent evt) : Action listener for images on the gameplay table.

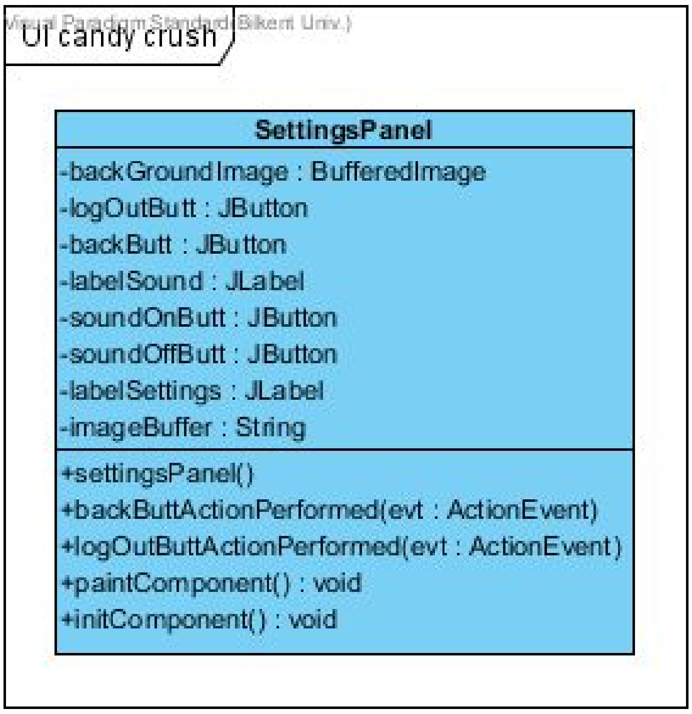
public void drawAnimation(Graphics g) : This method draws animations while crushing CS books.

public void drawSlideUp(Graphics g) : This method shows the animation while sliding horizontally.

public void drawSlideSide(Graphics g) : This method shows the animation while sliding vertically..

public void drawTable(Graphics g) : This method draws and redraws table images.

SettingsPanel Class



BufferedImage backGroundImage: The image to display in the background of SettingsPanel.

String imageBuffer: The directory to read the image to display in the SettingsPanel.

private javax.swing.JButton backButt: In settings, a JButton instance is called and it's modified to a button with "to back page" functionality.

private javax.swing.JButton logOutButt: : In settings, a JButton instance is called and it's modified to a button with "log user out" functionality.

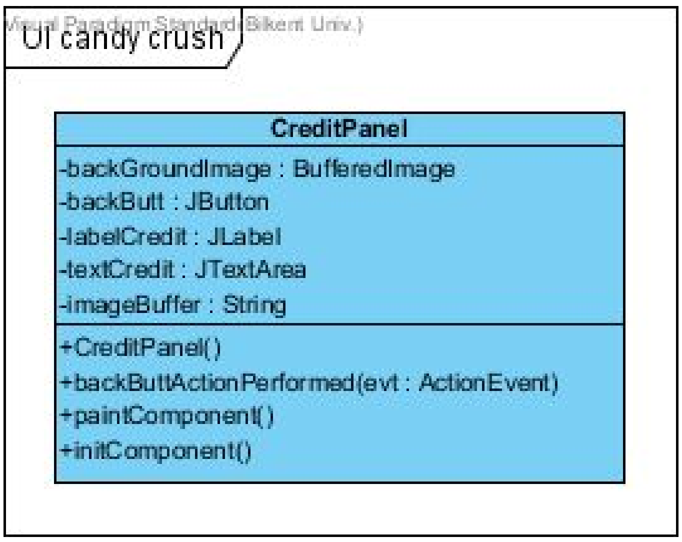
private javax.swing.JButton soundOnButt: : In settings, a JButton instance is called and it's modified to a button with "turn on sound"functionality.

private javax.swing.JButton soundOnButt: : In settings, a JButton instance is called and it's modified to a button with "turn off sound"functionality.

private javax.swing.JLable labelSound: A JLable instance for indicating sound options.

private javax.swing.JLable labelSettigs: A JLable instance for title of the settings page.

Credit Panel Class



BufferedImage backGroundImage: The image to display in the background of CreditsPanel.

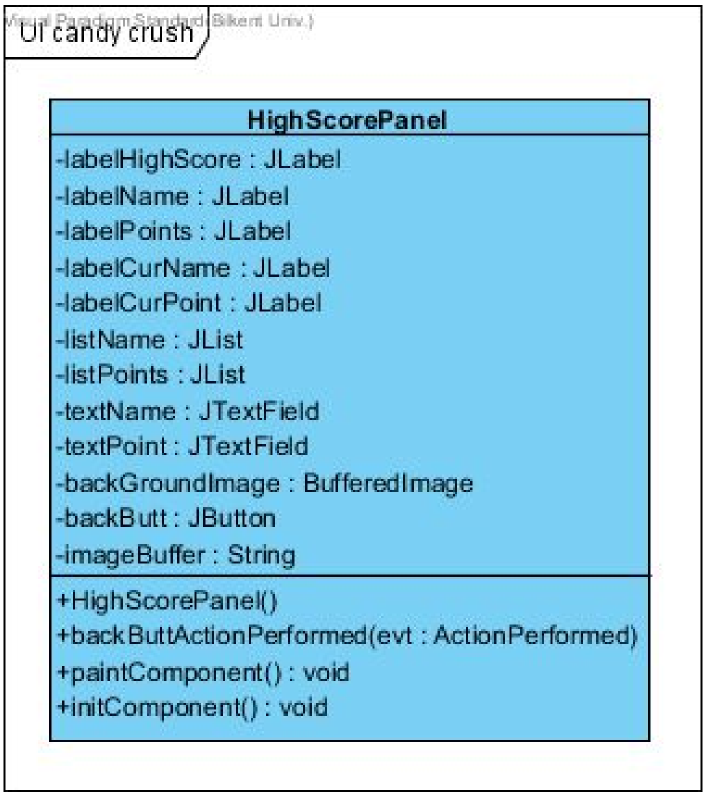
String imageBuffer: Address of image to be used is defined with this String attribute.

private javax.swing.JTextArea textCredit: This is a text area for developer names to appear.

private javax.swing.JButton backButt: This is a Button displayed in Credits panel which will be used to go back to main menu.

private javax.swing.JLabel labelCredit: This is the title string.

HighScore Panel Class



BufferedImage backGroundImage: The image to display in the background of HighScorePanel.

String imageBuffer : Address of image to be used is defined with this String attribute.

private javax.swing.JButton backButt : In HighScorePanel, a JButton instance is called and it's modified to a button with "to back page" functionality.

private javax.swing.JLabel labelHighScore : This is the panel where String imageBuffer will be printed.

private javax.swing.JLabel labelName : This is the panel where String imageBuffer will be printed.

private javax.swing.JLabel labelPoints : This is the panel where String imageBuffer will be printed.

private javax.swing.JLabel labelCurName : This is the panel where String imageBuffer will be printed.

private javax.swing.JLabel labelCurPoint : This is the panel where String imageBuffer will be printed.

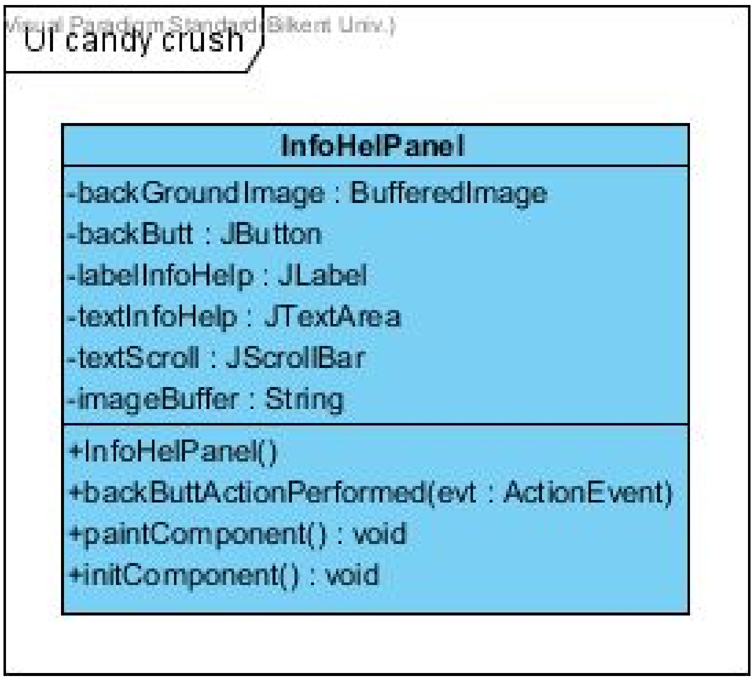
private javax.swing.JList listName : This is the list where scorers names will be printed.

private javax.swing.JList listPoints : This is the list where scorers points will be printed.

private javax.swing.JTextField textName : This is the area which will display the selected scrorer's name.

private javax.swing.JTextField textPoint : This is the area which will display the selected scrorer's points.

InfoHelp Panel Class



BufferedImage backgroundImage : The actual image to be taken with String attribute imageBuffer.

String imageBuffer : This string holds the address of image to be used as backgroudImage.

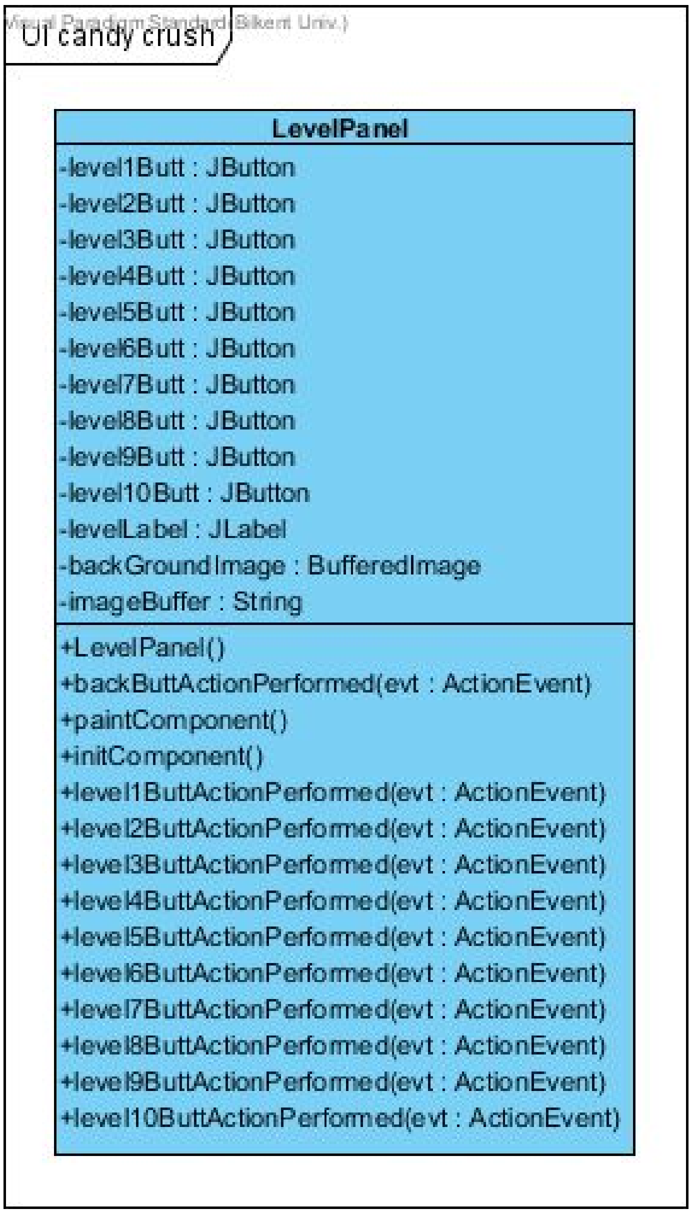
private javax.swing.JButton backButt : In HighScorePanel, a JButton instance is called and it's modified to a button with "to back page" functionality.

private javax.swing.JLabel labelName : This is the panel where String imageBuffer will be printed.

private javax.swing.JTextArea textInfoHelp : Text area where info and help instructions will be written.

private javax.swing.JScrollBar textScroll : Scroll bar for JtextArea.

Level Panel Class

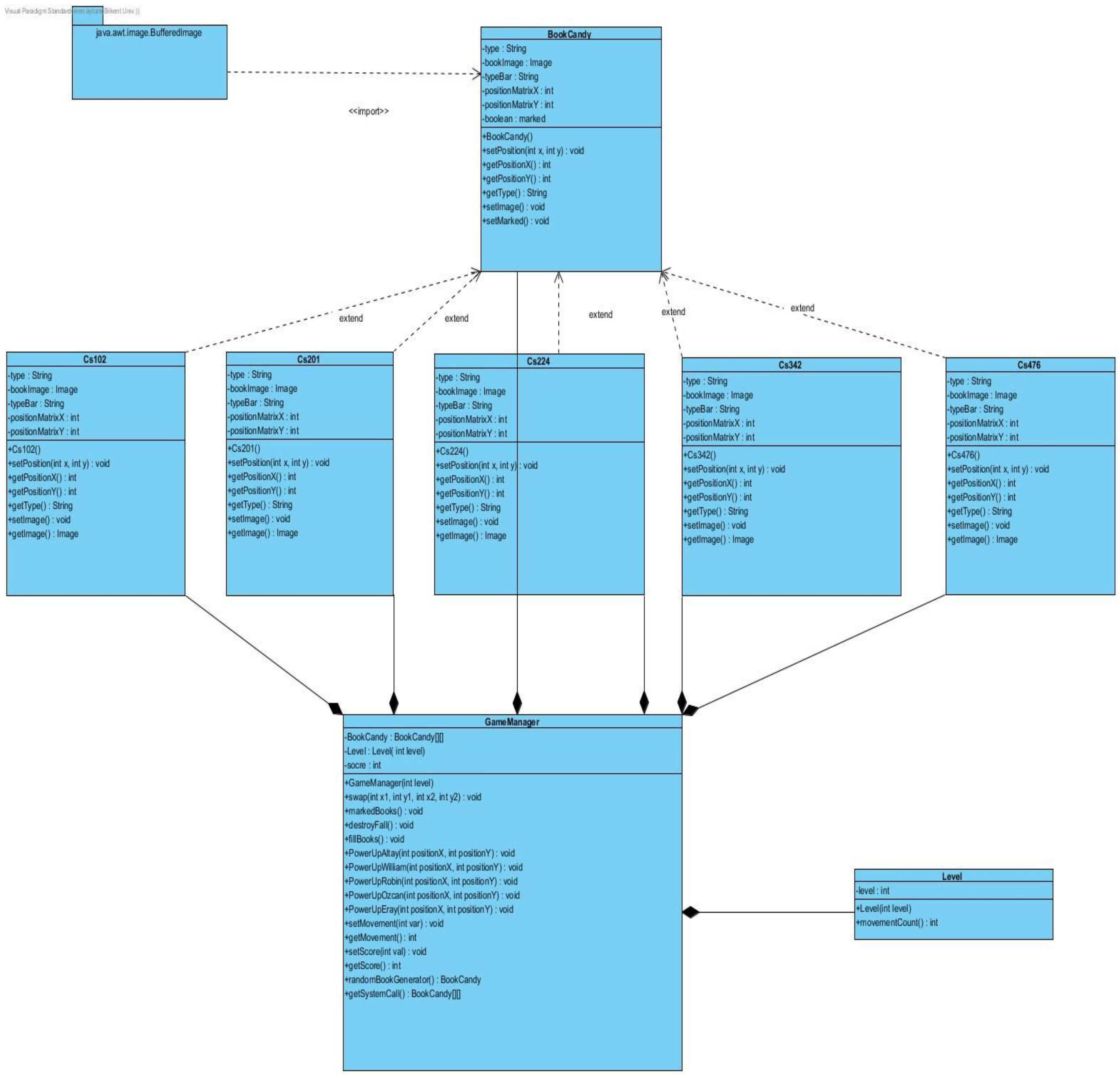

Visual Paradigm Standard (Bilkent Univ.)
UI candy crush
LevelPanel
-level1Butt : JButton
-level2Butt : JButton
-level3Butt : JButton
-level4Butt : JButton
-level5Butt : JButton
-level6Butt : JButton
-level7Butt : JButton
-level8Butt : JButton
-level9Butt : JButton
-level10Butt : JButton
-levelLabel : JLabel
-backGroundImage : BufferedImage
-imageBuffer : String
+LevelPanel()
+backButtActionPerformed(evt : ActionEvent)
+paintComponent()
+initComponent()
+level1ButtActionPerformed(evt : ActionEvent)
+level2ButtActionPerformed(evt : ActionEvent)
+level3ButtActionPerformed(evt : ActionEvent)
+level4ButtActionPerformed(evt : ActionEvent)
+level5ButtActionPerformed(evt : ActionEvent)
+level6ButtActionPerformed(evt : ActionEvent)
+level7ButtActionPerformed(evt : ActionEvent)
+level8ButtActionPerformed(evt : ActionEvent)
+level9ButtActionPerformed(evt : ActionEvent)
+level10ButtActionPerformed(evt : ActionEvent)

private javax.swing.JButton level(1-10)Butt : JButton instances which calls levels 1 to 10 respectively.

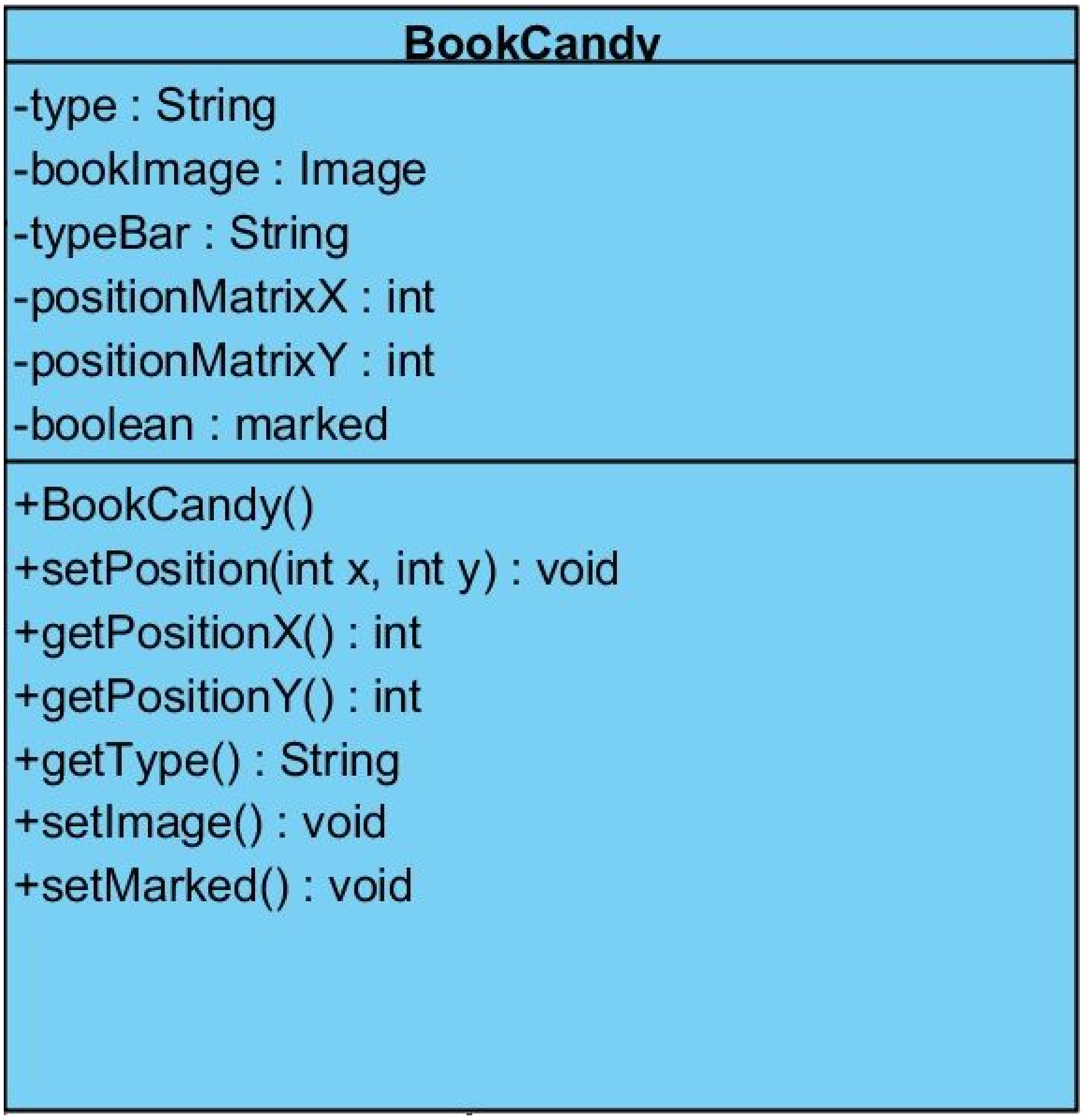
private javax.swing.JLabel levelLabel : This is the panel where String imageBuffer will be printed.

BufferedImage backgroundImage : The actual image to be taken with String attribute imageBuffer.

3.2.2 Game Logic Design And Subsystems



BookCandy Class



Constructor

+BookCandy() : This constructor is the parent of other constructors

Attributes

-private string type: According to this string type the program understands that the order of same types of book is provided or not.

-private Image bookImage: This image shows the type of books to the user.

-private string typeBar: In this game every book has a special type. These types are also splitted vertically or horizontally. This string determines that.

-private int PositionMatrixX: This integer type shows the x coordinate of position.

-private int PositionMatrixY: This integer type shows the y coordinate of position.

-private bool marked: The program understands that the book is special or no

Methods

+BookCandy(String barType):

+public void setPosition(int x, int y): According to swapping or randomly(while starting), this method sets the position of the books.

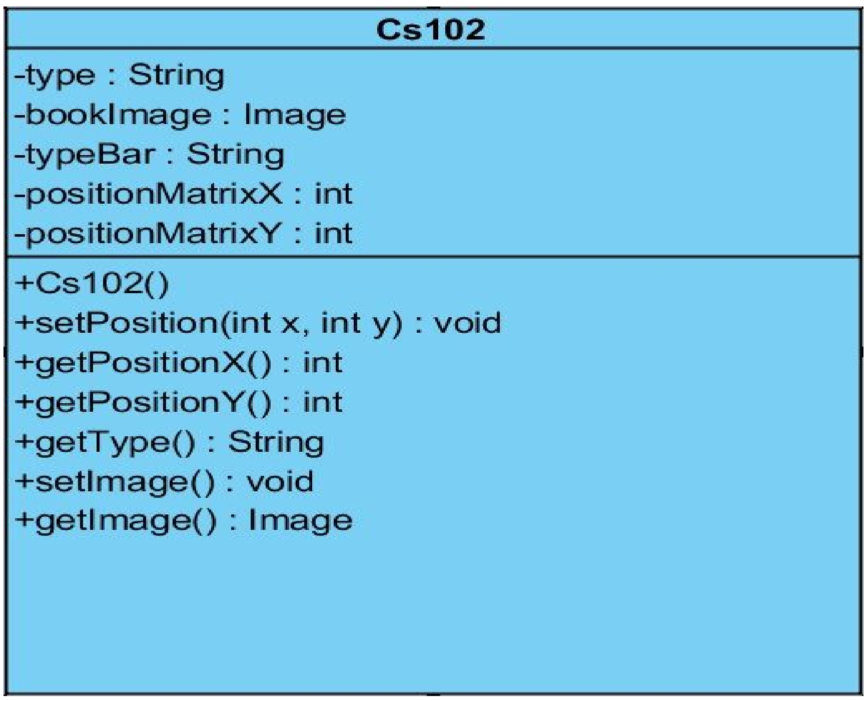
+public int getPositionX(): To disappear or swap the the books, the program should know the x coordinate of the books.

+public int getPositionY(): To disappear or swap the the books, the program should know the y coordinate of the books.

+public string getType(): The program understands the type of books according to the this string.

+public void setImage(): The icons of the books are arranged by using this method.

CS102 Class



```
Cs102
-type : String
-bookImage : Image
-typeBar : String
-positionMatrixX : int
-positionMatrixY : int
+Cs102()
+setPosition(int x, int y) : void
+getPositionX() : int
+getPositionY() : int
+getType() : String
+setImage() : void
+getImage() : Image
```

Constructor

+Cs102() : The type of the book can be created as a special type in the proportion of this is 5%. If it is special book, the bar type of this book is created as vertical or horizontal randomly.
If it is not special book, it is created normally. On the other hand, the type of this book is created randomly and bar type is arranged “None”.

Attributes

-private string type: The type of this string is is “cs102”.

-private Image bookImage: The icon of this book is “cs102.jpeg”

-private string typeBar: “Vertical”,”Horizontal” or ”None”

-private int PositionMatrixX: The program sets randomly the x coordinate of the book and it is changed by swapping.

-private int PositionMatrixY: The program sets randomly the x coordinate of the book and it is changed by swapping.

-private bool marked: If it is true, it is a special type of cs102 book.

Methods

+public void setPosition(int x, int y): According to swapping or randomly(while starting), this method sets the position of the cs 102 book.

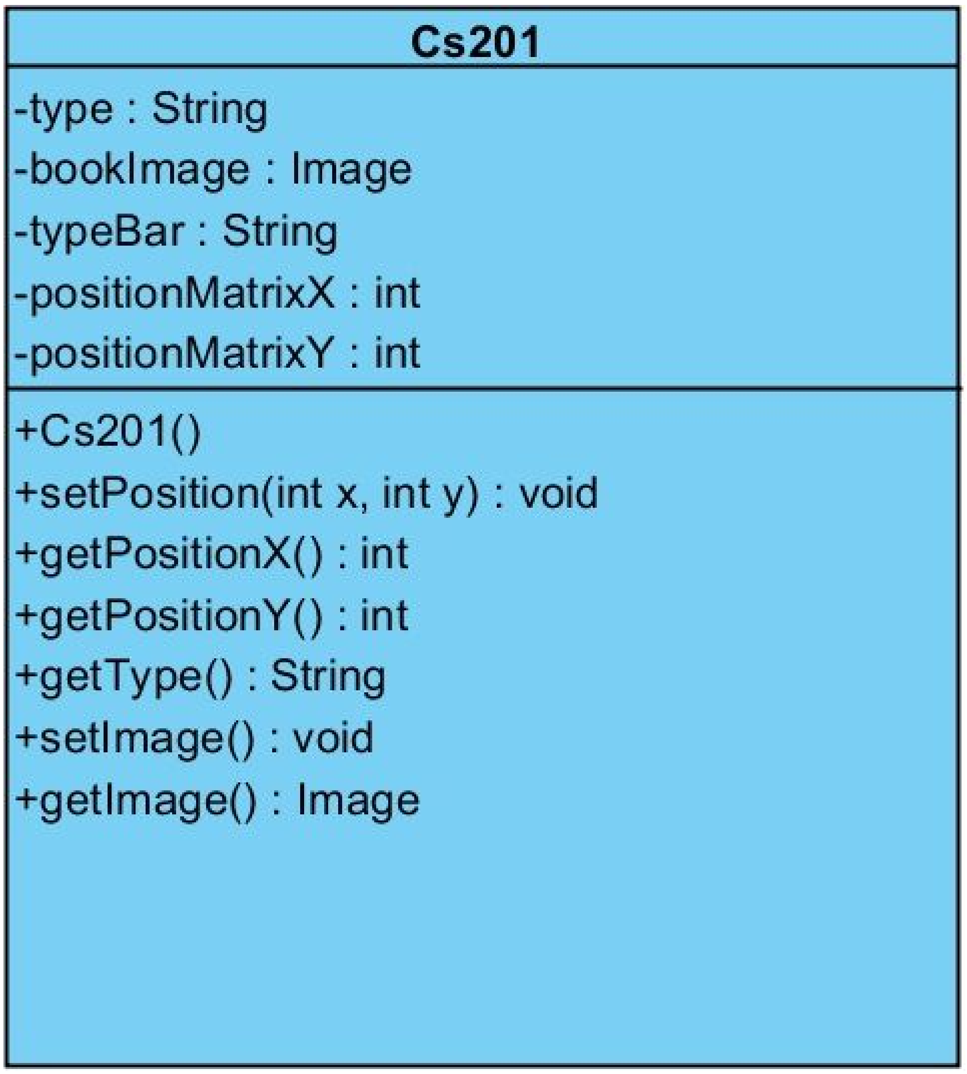
+public int getPositionX(): This method returns the x coordinate.

+public int getPositionY(): This method returns the y coordinate.

+public string getType(): This method returns the type of book as a string.

+public void setImage(): If the gamer uses power up, the type can be special type of the same book. This method changes the normal image with special image of the same book.

CS201 Class



Constructor

+Cs201() : The type of the book can be created as a special type in the proportion of this is 5%. If it is special book, the bar type of this book is created as vertical or horizontal randomly.

If it is not special book, it is created normally. On the other hand, the type of this book is created randomly and bar type is arranged “None”.

Attributes

-private string type: The type of this string is is “cs201”.

-private Image bookImage: The icon of this book is “cs102.jpeg”.

-private string typeBar: “Vertical”,”Horizontal” or ”None”

-private int PositionMatrixX: The program sets randomly the x coordinate of the book and it is changed by swapping.

-private int PositionMatrixY: The program sets randomly the y coordinate of the book and it is changed by swapping.

-private bool marked: If it is true, it is a special type of cs102 book.

Methods

+public void setPosition(int x, int y): According to swapping or randomly(while starting), this method sets the position of the cs 201 book.

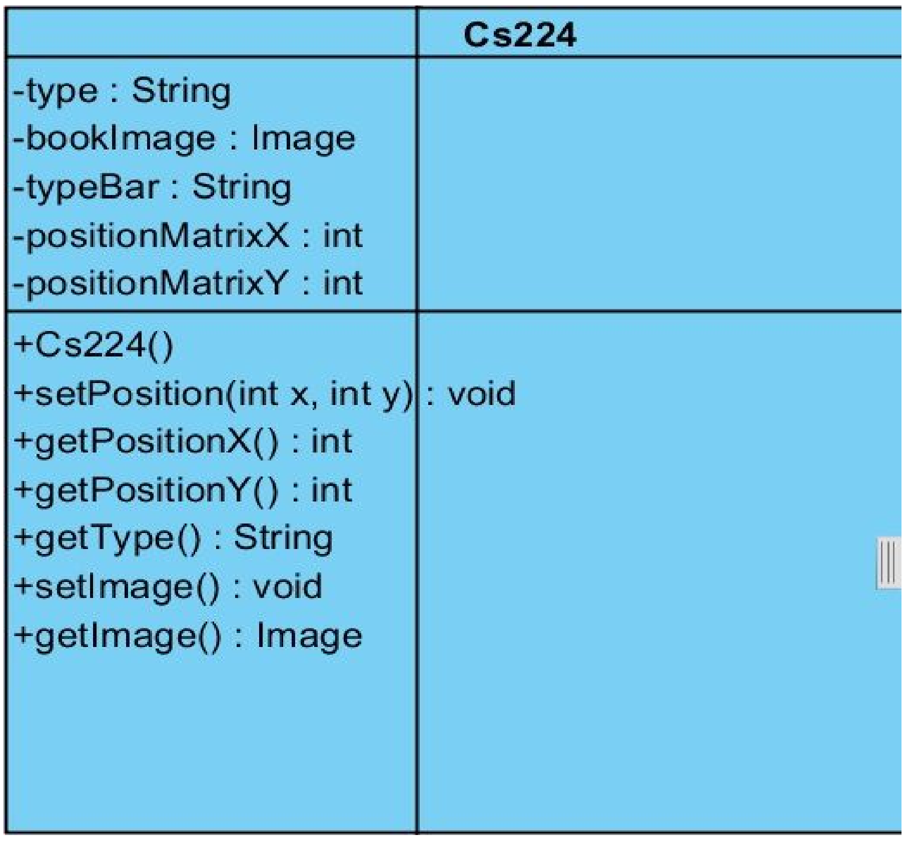
+public int getPositionX(): This method returns the x coordinate.

+public int getPositionY(): This method returns the y coordinate.

+public string getType(): This method returns the type of book as a string.

+public void setImage(): If the gamer uses power up, the type can be special type of the same book. This method changes the normal image with special image of the same book.

CS224 Class



Constructor

+Cs224() : The type of the book can be created as a special type in the proportion of this is 5%. If it is special book, the bar type of this book is created as vertical or horizontal randomly.
If it is not special book, it is created normally. On the other hand, the type of this book is created randomly and bar type is arranged “None”.

Attributes

-private string type: The type of this string is is “cs224”.

-private Image bookImage: The icon of this book is “cs224.jpeg”.

-private string typeBar:

-private int PositionMatrixX: The program sets randomly the x coordinate of the book and it is changed by swapping.Before swapping, the game determines the x coordinate of the book by starting.

-private int PositionMatrixY: The program sets randomly the y coordinate of the book and it is changed by swapping.Before swapping, the game determines the y coordinate of the book by starting.

-private bool marked: If it is true, it is a special type of cs224 book.

Methods

+public void setPosition(int x, int y): According to swapping or randomly(while starting), this method sets the position of the cs 224 book.

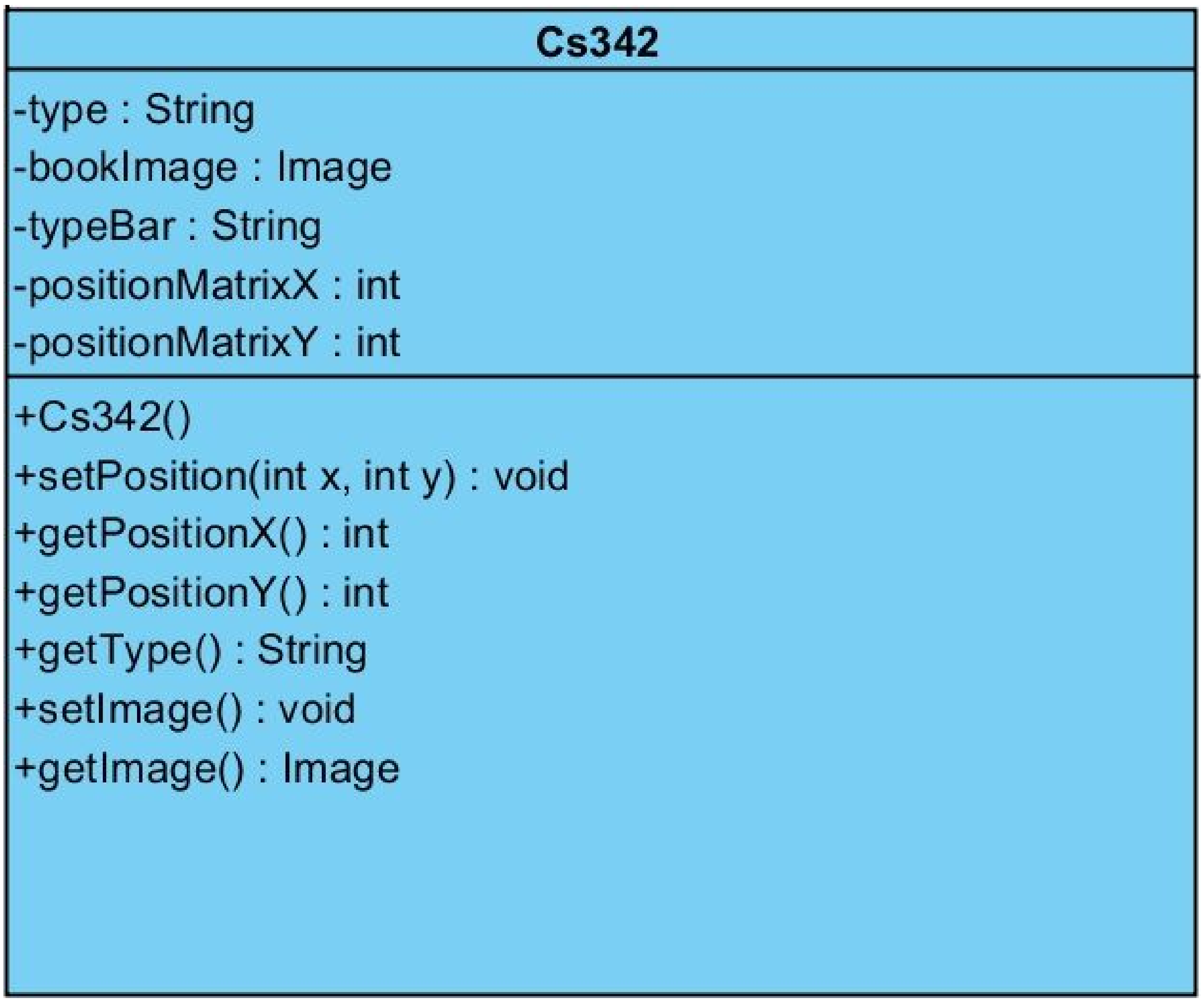
+public int getPositionX(): This method returns the x coordinate.

+public int getPositionY(): This method returns the y coordinate.

+public string getType(): This method returns the type of book as a string.

+public void setImage(): If the gamer uses power up, the type can be special type of the same book. This method changes the normal image with special image of the same book.

CS342 Class



Constructor

+Cs342() : The type of the book can be created as a special type in the proportion of this is 5%. If it is special book, the bar type of this book is created as vertical or horizontal randomly.

If it is not special book, it is created normally. On the other hand, the type of this book is created randomly and bar type is arranged “None”.

Attributes

-private string type: The type of this string is is “cs342”.

-private Image bookImage: The icon of this book is “cs342.jpeg”.

-private string typeBar:“Vertical”,”Horizontal” or ”None”

-private int PositionMatrixX: The program sets randomly the x coordinate of the book and it is changed by swapping.Before swapping, the game determines the x coordinate of the book by starting.

-private int PositionMatrixY: The program sets randomly the y coordinate of the book and it is changed by swapping.Before swapping, the game determines the y coordinate of the book by starting.

-private bool marked: If it is true, it is a special type of cs342 book.

Methods

+public void setPosition(int x, int y): According to swapping or randomly(while starting), this method sets the position of the cs 342 book.

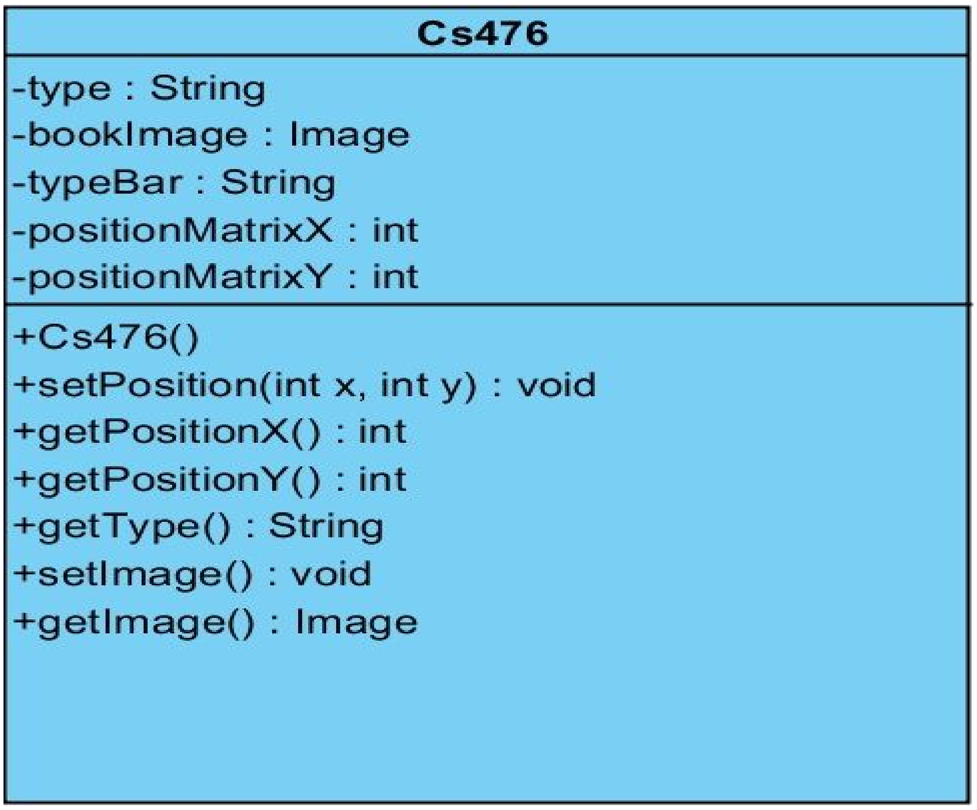
+public int getPositionX(): This method returns the x coordinate.

+public int getPositionY(): This method returns the y coordinate.

+public string getType(): This method returns the type of book as a string.

+public void setImage(): If the gamer uses power up, the type can be special type of the same book. This method changes the normal image with special image of the same book.

CS476 Class



Constructor

+Cs476() : The type of the book can be created as a special type in the proportion of this is 5%. If it is special book, the bar type of this book is created as vertical or horizontal randomly.
If it is not special book, it is created normally. On the other hand, the type of this book is created randomly and bar type is arranged “None”.

Attributes

-private string type: The type of this string is is “cs476”.

-private Image bookImage: The icon of this book is “cs476.jpeg”.

-private string typeBar:

-private int PositionMatrixX: The program sets randomly the x coordinate of the book and it is changed by swapping. Before swapping, the game determines the x coordinate of the book by starting.

-private int PositionMatrixY: The program sets randomly the y coordinate of the book and it is changed by swapping. Before swapping, the game determines the y coordinate of the book by starting.

-private bool marked: If it is true, it is a special type of cs book.

Methods

+public void setPosition(int x, int y): According to swapping or randomly(while starting), this method sets the position of the cs 476 book.

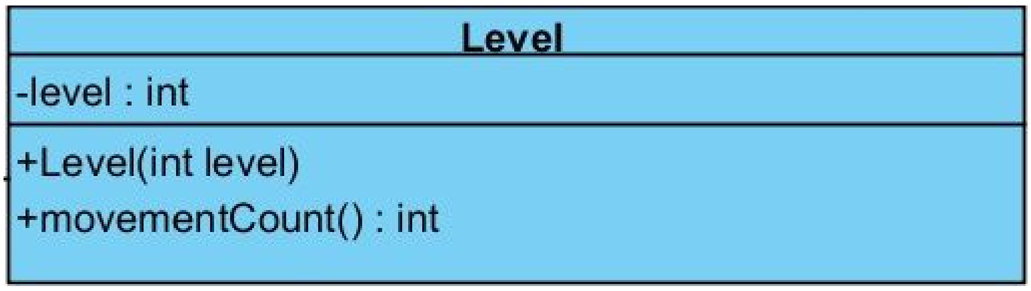
+public int getPositionX(): This method returns the x coordinate.

+public int getPositionY(): This method returns the y coordinate.

+public string getType(): This method returns the type of book as a string.

+public void setImage(): If the gamer uses power up, the type can be special type of the same book. This method changes the normal image with special image of the same book.

Level Class



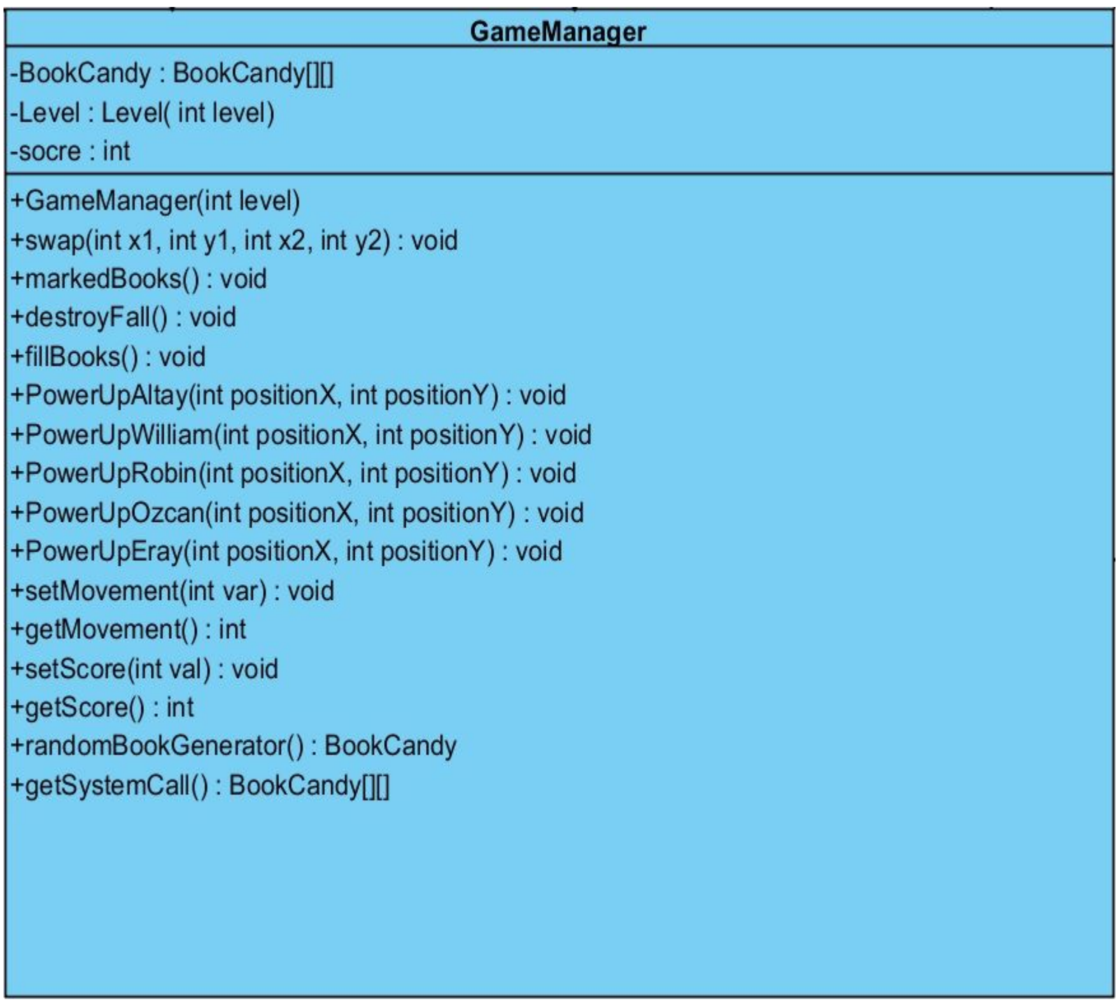
Attributes

-private int level: Determines the level 1 to 10

Methods

+public int movementCount(): Generates movement counts according to level.

GameManager Class


GameManager
-BookCandy : BookCandy[][]
-Level : Level(int level)
-socre : int
+GameManager(int level)
+swap(int x1, int y1, int x2, int y2) : void
+markedBooks() : void
+destroyFall() : void
+fillBooks() : void
+PowerUpAltay(int positionX, int positionY) : void
+PowerUpWilliam(int positionX, int positionY) : void
+PowerUpRobin(int positionX, int positionY) : void
+PowerUpOzcan(int positionX, int positionY) : void
+PowerUpEray(int positionX, int positionY) : void
+setMovement(int var) : void
+getMovement() : int
+setScore(int val) : void
+getScore() : int
+randomBookGenerator() : BookCandy
+getSystemCall() : BookCandy[][]

Constructor

BookCandy 2D array is filled by randomBookGenerator() function.

Attributes

-private bookCandy[][]: This array is going to be used for representing the matrix because all visual operations will be checked from this 2D array. This array will hold child classes of BookCandy class. This array will be 10x10 matrix.

-private Level level: This object will generate movement count according to the 1 to 10 level.

-private int score: holds the score

Methods

+public void swap(): This method gives permission to swap objects in the margin and 1 next to up,right,left and down(0<=(position+10,position-1,position+1,position-10)<100)).This method calls markedBooks() method, then calls destroyFall() method and fillBooks() Method.

+private void markedBooks(): This method checks if 3 same book either horizontal or vertically and straightly together, then marks them to be destroyed.

+private void destroyFall(): This method checks marked objects in the array then sets null. Null Positions are filled with position-10k(0<k<10) existing objects in the margin of the matrix.

+private void fillBooks(): Null objects are filled with randomBookGenerator() and it stops if loop end or one raw is not filled with anything. This method again calls markedBooks() method, then calls destroyFall() method and fillBooks() method until no object is marked.

+public void PowerUpAltay(int positionX, int positionY): Turns 5 random objects into barType objects.

+public void PowerWilliam(int positionX, int positionY): Inside the 10x10 matrix,8x8 matrix objects can be chosen because this method set these objects and 3x3 matrix is set to null.

+public void PowerUpRobin(int positionX, int positionY): Regenerates a selected position in matrix.

+public void PowerUpOzcan(int positionX, int positionY): Sets one object marked and sets null.After that this method calls, then calls destroyFall() method and fillBooks() Method.

+public void PowerUpEray(int positionX, int positionY): This method can swap any book objects and calls, then calls destroyFall() method and fillBooks() Method.

+public void setMovement(int val): Shifts movement count.

+public int getMovement(): Returns number of movements.

+public BookCandy randomBookGenerator(): Generates books with its type randomly.

+public BookCandy[][] getSystemCall(): Returns 2D BookCandy objects array.