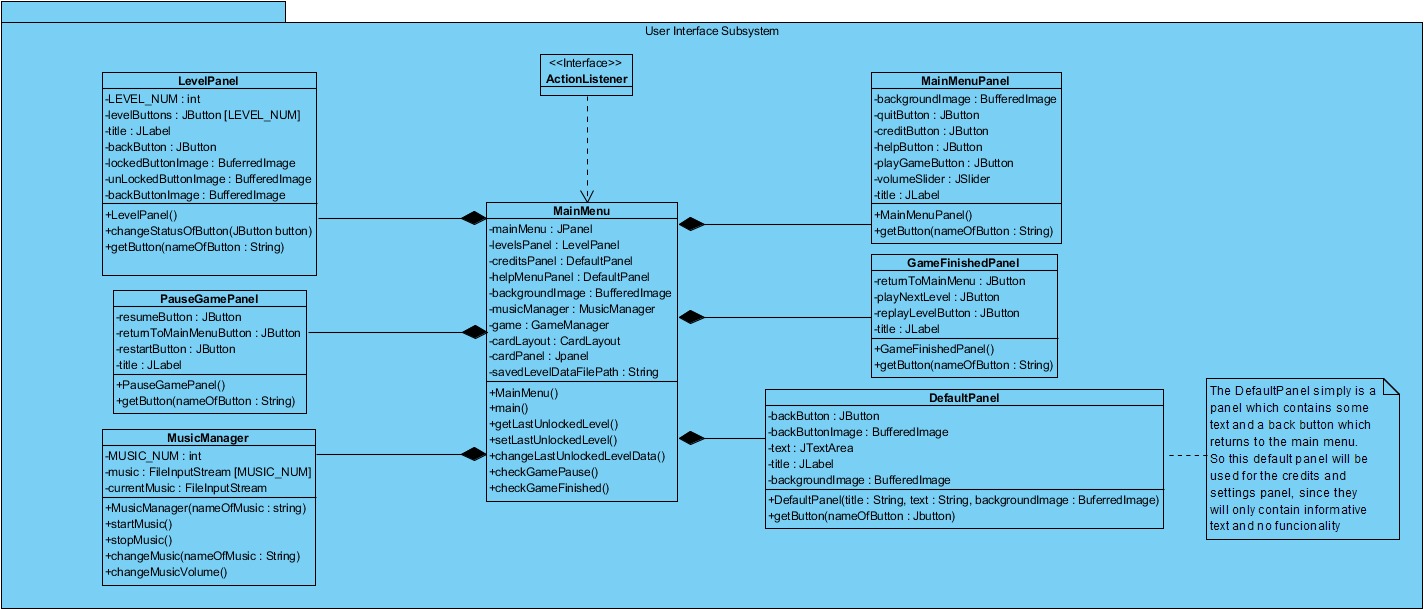
**User Interface Subsystem:**

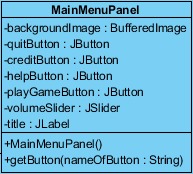
This subsystem includes the classes which have the features of the interface of the game.

It includes 7 classes which are, MainMenuPanel, LevelPanel, DefaultPanel, GameFinishedPanel, PauseGamePanel, MusicManager and MainMenu. MainMenu is the class that uses all the panels and MusicManager, and the GameManager to initiate the game. The ActionListener interface is to proceed the game according to the clicked buttons.



**MainMenuPanel Class:**

MainMenuPanel is a panel which include 4 buttons and a slider. First button is helpButton to be placed at the upper left corner of the screen and it opens a defaultPanel including a text about the description of the game. Second button is playGameButton, which leads to the levelsPanel. Third button is creditButton and it shows the developers of the game in a defaultPanel. Fourth button is quitButton and when it is clicked, game is immediately quit. playGameButton, creditButton and quitButton will be placed in the middle of the screen respectively. The slider is placed at the upper left corner of the screen. The player will be able to change the volume of the bacgground music by scrolling the volumeSlider. MainMenuPanel will provide the user interface of the first screen to appear when the game is clicked.

****

**Attributes:**

* **backgroundImage:** this attribute is a BufferedImage to be shown on the background of the main menu.
* **quitButton:** quitButton is one of the JButtons to be available on the main menu panel, which will be put on the bottom of the page. When it is called, the game will be quit.
* **creditButton:** creditButton is a JButton and when it is clicked, the developers of the game will be displayed on the screen.
* **helpButton:** helpButton is a JButton which will open a DefaultPanel including text of information and guidance for the game, when it is clicked.
* **playGameButton:** is a JButton. When it is clicked, LevelsPanel will appear.
* **volumeSlider:** is a JSlider to be placed at the upper left corner of the screen and will enable the player increase/decrease the volume palying on the background, which is called from MusicManager class.
* **title:** is a JLabel that includes the name of this panel.

**Constructor:**

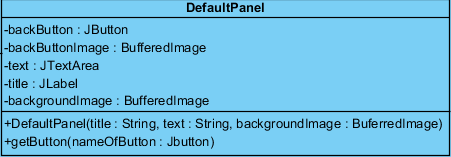
* **MainMenuPanel():** MainMenuPanel() is the default constructor of this class. It takes and sets the backgroundImage; creates playGameButton, creditButton, quitButton, helpButton and volumeSlider on the page and place them on the panel, setting their dimensions and sets the related title to the panel.

**Methods:**

* **getButton(nameOfButton : String):** getButton takes the button of this panel as a JButton parameter, called nameOfButton. It is then used by ActionListener to understand if the button clicked by the player is the button of this panel or another panel. After ActionListener decides which panel’s button is clicked by checking also the other panels’ getButton methods, it proceeds the related panel.

**DefaultPanel Class:**

DefaultPanel is simply a panel including a text and backButton which will enable the user get back to the main menu. DefaultPanel will be used as credits and help panels because they only have texts but no functionalities. Because other panels such as MainMenuPanel and PausePanel, have their own functional features, they are separated from DefaultPanel.



**Attributes:**

* **backButton:** backButton is a JButton that will help the user go to the previous page when it is clicked.
* **backButtonImage:** is a BufferedImage. It will be placed on the backButton.
* **text:** text is a JTextArea that will include some text depending on where it is called.
* **title:** title is a JLabel which will include the name of the current page.
* **backgroundImage:** is a BufferedImage. It will be placed on the backButton.

**Constructor:**

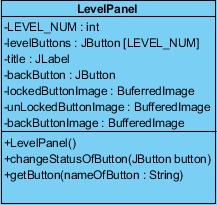
* **DefaultPanel(title: String, text: String, backgroundImage: BufferedImage):** this constructor creates a simple panel having a title on the page, a background image and a text. It takes the title, text and background image of the current page as its parameters, where title and text are String and backgroundImage is a BufferedImage.

**Methods:**

* **getButton(nameOfButton : JButton):** getButton takes the button of this panel as a JButton parameter, called nameOfButton. It is then used by ActionListener to understand if the button clicked by the player is the button of this panel or another panel. After ActionListener decides which panel’s button is clicked by checking also the other panels’ getButton methods, it proceeds the related panel.

**LevelPanel Class:**

LevelPanel is a panel having 5 buttons and title indicating it is the levels page. One of the buttons is backButton, which is to be available on every panel and returns to main menu. Other buttons correspond to 4 different levels. Each button will have an image on them showing whether that level is locked or unlocked. If the level is unlocked, the button will be clickable, or else, it will not be clickable. At the very beginning, only the first level’s button will be clickable and have unlockedButtonImage on it. Afterwards, as the new levels are completed successfully, their status of clickability will change by changeStatusOfButton method. As an unlocked level button is clicked, the game will be started by GameManager from the chosen level.



**Attributes:**

* **LEVEL\_NUM:** is an integer for the number of levels. LEVEL\_NUM will take 4 different values like 1, 2, 3 and 4, each corresponding to different levels.
* **levelButtons:** levelButtons is an array of JButton with size of LEVEL\_NUM. Each button is for different levels and the game starts the level when any of these levelButtons is clicked if unlocked.
* **title:** title is a JLabel including the name of the panel.
* **backButton:** is the JButton to help the user go to the previous page.
* **backButtonImage:** is a BufferedImage. It will be placed on the backButton.
* **lockedButtonImage:** is a BufferedImage which will be placed on the levelButtons which are locked.
* **unLockedButtonImage:** is a BufferedImage which will be placed on the levelButtons which are unlocked.

**Constructor:**

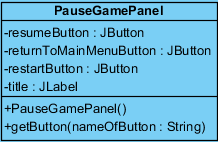
* **LevelPanel(): is the** default constructor. It creates the buttons and put them on their places accordingly; sets button colors and images, sets background image and sets the title of the panel.

**Methods:**

* **changeStatusOfButton(JButton button):** Each level button have a status, either locked or unlocked. The unlocked levels’ buttons will not clickable and have lockedButtonImage on. Therefore, if a level is finished successfully, the button of the next level will become clickable and its image will be changed to unlockedButtonImage from lockedButtonImage. changeStatusOfButton takes a JButton, called button for this method, whose current status is needed to be changed as a parameter.
* **getButton(nameOfButton : Jbutton):** getButton takes the button of this panel as a JButton parameter, called nameOfButton. It is then used by ActionListener to understand if the button clicked by the player is the button of this panel or another panel. After ActionListener decides which panel’s button is clicked by checking also the other panels’ getButton methods, it proceeds the related panel.

**PauseGamePanel Class:**

PauseGamePanel is a panel which will appear when the player clicks “P” during the game. It appears as a pop-up and includes resumeButton, returnToMainMenuButton, restartButton and a title indicating the game is paused. When resumeButton is clicked, game continues from where it is left. When returnToMainMenuButton is clicked, game finishes and MainMenuPanel is open. When the restartButton is clicked, same level restarts immediately.

****

**Attributes:**

* **resumeButton:** resumeButton is a JButton that returns the game from the pause panel.
* **returnToMainMenuButton:** is a JButton helping the user return to main menu page.
* **restartButton:** is a JButton which restarts the current level.
* **title**: is a JLabel for the title of the Pause page.

**Constructor:**

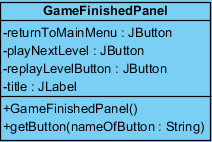
* **PauseGamePanel():** is the default constructor. It creates a pop-up including a title indicating the game is paused, restarts button, resumeButton and returnToMainMenuButton and sets the buttons clickable.

**Methods:**

* **getButton(nameOfButton: String):** getButton takes the button of this panel as a JButton parameter, called nameOfButton. It is then used by ActionListener to understand if the button clicked by the player is the button of this panel or another panel. After ActionListener decides which panel’s button is clicked by checking also the other panels’ getButton methods, it proceeds the related panel.

**GameFinishedPanel Class:**

GameFinishedPanel is a panel that displays when the level is finished successfully. It appears as a pop-up like PauseGamePanel and includes replayLevelButton, returnToMainMenu, playNextLevel buttons and a title indicating the game is finished successfully. When the button returnToMainMenu is clicked, MainMenuPanel appears. When playNextLevel button is clicked, the next level of the game starts immediately. If the user clicks replayLevelButton, the next level be unlocked, which means the success is recorded, but the same level will be started again instead of the new level .

****

**Attributes:**

* **returnToMainMenu:** is a JButton for the user to go back to the beginning page, which is the MainMenu.
* **playNextLevel:** is a JButton to pass the next level when the level is completed successfully.
* **replayLevelButton:** is a JButton which enables the user to play the same level again.
* **title:** is aJLabel for the title of GameFinishedPanel to be shown when the game is over.

**Constructor:**

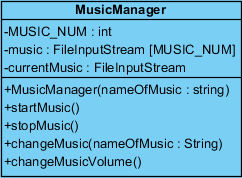
**GameFinishedPanel():** is the default constructor of GameFinishedPanel class. It shows up as a pop-up on the game. It creates a title indicating the game is finished and creates replayLevelButton, returnToMainMenu and playNextLevel buttons, and set the buttons clickable.

**Methods:**

* **getButton(nameOfButton: String):** getButton takes the button of this panel as a JButton parameter, called nameOfButton. It is then used by ActionListener to understand if the button clicked by the player is the button of this panel or another panel. After ActionListener decides which panel’s button is clicked by checking also the other panels’ getButton methods, it proceeds the related action.

**MusicManager Class:**

During the game, music will play on the background and the MusicManager is to manage the music related part of the game. As the game clicked and MainMenuPanel is open, music starts. The same music continues as LevelsPanel, PauseGamePanel and GameFinishedPanel are open. Once the game starts, the music changes. There will be some different sound effects, which are also taken as music in the input file, that will start as the dot jumps, falls, collides etc. The player will not be able to start or stop the music, but s/he can only make it silent by changing the volume.

****

**Attributes:**

* **MUSIC\_NUM:** is an integer. MUSIC\_NUM will correspond to a music to be displayed on the background of the game in the music file.
* **music:** is a JButton to pass the next level when the level is completed successfully.
* **currentMusic:** is a JButton which enables the user to play the same level again.

**Constructor:**

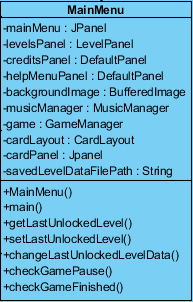
* **musicManager(nameOfMusic : String):** The constructor of MusicManager class takes the name of the music to be played on the background as a String parameter, named nameOfMusic. It sets the nameOfMusic as currentMusic.

**Methods:**

* **startMusic():** startMusic makes a music start to play. When the corresponding MUSIC\_NUM is used in music fileInputStream array, that music starts. currentMusic will hold the currently playing music and as the music changes, currentMusic will shift to the new music.
* **stopMusic():** stopMusic method makes the currentMusic stop playing. It will be used to stop the music when a new music is to be started to play.
* **changeMusic(nameOfMusic: String):** changeMusic changes the currentMusic by starting(startMusic) a new music and stopping(stopMusic) the last one. This method is to be used to change the music as the game starts, or a new panel is opened during the game. For example, while currentMusic is music1 during the game, if the game is paused and PausePanel appear, it will change to music2. The new music which is wanted to play after changeMusic method will be taken as a String parameter holding the name of that music in the input file, which is nameOfMusic: String.
* **changeMusicVolume():** changeMusicVolume() method controls the colume of the currentMusic as the player uses the volumeSlider. If the player slides it left side, volume will decrease; else if s/he slides it to the right, volume will increase. If the user does not use volumeSlider and stop sliding, the volume will stay same where the slide is left.

**MainMenu Class:**

MainMenu class uses MainMenuPanel, LevelsPanel and DefaultPanel. It also uses GameManager to initiate the game. According to the flow of the game, it decides which panel is to be open, which methods to be called, which data to be saved etc.

****

**Attributes:**

* **mainMenu:** mainMenu is a MainMenuPanel that will display the Main Menu contents.
* **levelsPanel:** levelsPanel is a LevelPanel which is a separate panel from mainMenu. It includes the features provided by LevelPalen class.
* **creditsLayout:** creditsLayout is a DefaultPanel, having the features of DefaultPanel class. In this panel, brief information about the developers of “Run Dot Run” will be shown as a text**.**
* **helpMenuPanel:** helpMenuPanel is also a DefaultPanel, which will include a text of description of the game.
* **backgroundImage:** backgroundImage is a BufferedImage which will be seen on the background of the game.
* **musicManager:** musicManager is a MusicManager to control the music playing on the background of the game depending on which panel is open.
* **cardLayout:** is a CardLayout to provide the transition among all panels of the game, which are all put in one frame. It is put into cardPanel, which is a Jpanel.
* **cardPanel:** is a JPanel where all the panels (MainMenuPanel, LevelPanel, DefaultPanel etc.) are added into. It shifts the panels as their buttons are clicked, or any other necessary condition is provided to open that panel.
* **savedLevelDataFilePath:** this attribute is to hold a string for the lastly saved level’s name.
* **game:** is a GameManager as the reference to the GameManager class.

**Constructor:**

* **MainMenu():** is the default constructor.

**Methods:**

* **main():** is the method to initiate the game.
* **setLastUnLockedLevel():** this method sets the last unlocked level to keep which level the game is left and up to which level the others are unlocked.
* **getLastUnLockedLevel():** gives the last unlocked level.
* **checkGamePause():** returns true if the game is paused, and opens PauseGame panel.
* **checkGameFinished():** checks if the game is over and opens gameFinished panel.