Design Document

**System Name:**

Application.java

**System Description:**

The program runs a simple side-scrolling game.

**Requirements:**

The game is to support:

* A basic GUI
* multiple stationary and moving objects
* multiple levels with different objects
* A highscores list that maintains the top 10 high scores
* A menu bar to support various options such as pause, stop, exit, new game etc

**Class Descriptions**

|  |
| --- |
| Application:  The main class of the project. It extends the JFrame class and implements the ActionListener and KeyListener interfaces, and is responsible for holding the GUI. It is the main class. |
| Instance variables:   |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | game | Game | A panel to hold the scores, game and main screens. | | t | Timer | A clock to keep things in the game moving. | | mb | MenuBar | A component to interact with the user and receive input, such as a request to start a new game. | |
| Member methods:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Description** | **Pre/post conditions** | **Interaction** | **I/O** | | Application() | Constructor to instantiate objects. | N/A | Calls the constructors of the various member objects,  Sets up the menu bar. | None | | void main() | The main method of the class. | N/A | Calls the class constructor, adds the various gui components, starts the timer. | None | | void actionPerformed  (ActionEvent) | Listens for timer firings | Timer must fire. | Advances the game by a tick, repaints all components. | Displays components on the monitor. | | void keyTyped  (KeyEvent) | Listens for key entries | A key must be typed. | Calls the games input(char) method, asking it to handle the new input. | None. | |

|  |
| --- |
| Game:  Extends the JPanel class, and holds the menu, game and score panels in a CardLayout, displaying the appropriate one on type based on current state. |
| Instance variables:   |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | menu | menuPanel | A panel to hold and display the start screen. | | game | gamePanel | A panel to hold and display the game. | | scores | scoresPanel | A panel to hold and display the scores. | | cl | CardLayout | Layout Manager, used to show a certain panel on top. | |
| Member methods:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Description** | **Pre/post conditions** | **Interaction** | **I/O** | | Game  (CardLayout cl) | Constructor to instantiate objects. | N/A | Adds the various components to the panel, specifying the layout manager. | Puts the start screen panel on top. | | void back() | Method to return to the start screen from the game or the highscores view. | N/A | Calls the ObjectOutput’s read() and close() methods to save the game to the filesystem. | Brings the appropriate card to the top. | | void load() | Loads a previously saved game | The requested game must exist in the filesystem. | Uses the ObjectInputStream class’ readObject() methods to deserialize the file and load it as a game into memory. | Uses a JDialog to request the name.  Displays the loaded game. | | void input(char) | Decides what to do next based on the value of char:  ‘0‘advances the game  ‘n’ starts a new game  ‘h’ displays the scores  ‘p’ pauses the game. | An option must be selected from the menu bar or a key must be typed. | Uses the game.move(char) method to pass inputs to the actual game. | Brings the appropriate card to the front. | |

|  |
| --- |
| MenuPanel:  A very basic class to hold a start screen that instructs the user on how to begin. Extends the JPanel class. |
| Instance variables:   |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | menu | JLabel | A simple label that holds instructions. | |
| Member methods:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Description** | **Pre/post conditions** | **Interaction** | **I/O** | | MenuPanel() | Constructor that simply adds the JLabel to the panel. | N/A | None | None | |

|  |
| --- |
| ScoresPanel:  Extends the JPanel class. It stores and displays the scores of the players. It also includes a Score class that holds name-score pairs, and implements the Comparable interface for comparision and the Serializable interface so that it can be saved for retrieval later. |
| Instance variables:   |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | queue | PriorityQueue<Score> | Stores the scores, highest first. | | out | JLabel | A label to display the scores. | | callerstate | int | Used to determine whether to return to the game or the start screen. | | outtext | String | Holds the text to be used in the JLabel. | |
| Member methods:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Description** | **Pre/post conditions** | **Interaction** | **I/O** | | ScoresPanel() | Constructor to instantiate objects. | If a ‘scores.save’ file exists, it must hold a valid PriorityQueue of Score objects. | Uses the ObjectInputStream class to deserialize an existing score record, if it already exists. Uses the refresh() method to keep out up-to-date. | If the file ‘scores.save’ exists, loads it to memory. Otherwise, creates a new file. | | void addScore(int, String) | Method that allows addition of entries. | The queue must stay sorted . | Uses queue’s add() method to add the entry. Also uses the refresh() method. | None | | void refresh() | To keep out updated. | out must display the latest scores. | Uses the Iterator supplied by the PriorityQueue to extract score data. | Updates out. | |

|  |
| --- |
| Game:  Extends the JPanel class. This is a container that holds the actual game display, including the current score, the player and the other game objects. |
| Instance variables:   |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | parent | Game | Reference to the Game object that creates this object. | | paused | boolean | Tells the game whether or not to advance. | | next | int | Counter to keep track of the number of game ticks. | | startx | int | Holds the starting point from which to draw objects. | | p | Player | Holds the player. | | objects | ArrayList <GameObject> | Holds the other objects in the game. | | name | String | Stores the player’s name | | score | JLabel | Displays the user’s current score. | | ctr | int | Counter to store the number of times the player has moved | |
| Member methods:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Description** | **Pre/post conditions** | **Interaction** | **I/O** | | GamePanel() | Constructor to initialize the world. | None | Uses the add() function of the ArrayList to place new objects. | None | | void restore(Game) | Reinitializes the filepaths of the object images. | Only called after deserializing this object. | Calls the restoreImg() methods of p and the elements in objects. | None | | void move() | To handle player motion. | None | Calls the up(), down() or right() methods of p as appropriate. | None | | void next() | Advances the game by a tick, detects collisions and updates the score. | None | Uses the iterator provided by the ArrayList to traverse the list of gameobjects. Also uses the getRect() method of the Player and GameObject classes to detect collisions. | Uses a JOptionPane to inform the player that the game is over. | | void paint() | Paints the objects on the map. | None | Uses the getImage() method of the Player and GameObject classes to find their image files. | Paints the map to the panel. | |

|  |
| --- |
| Player:  Holds the player’s character. |
| Instance variables:   |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | img | Image | Stores thesprite. | | x | int | Holds the coordinates. | | y | | lives | int | Holds the number of remaining lives. | | score | int | Holds the current score. | |
| Member methods:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Description** | **Pre/post conditions** | **Interaction** | **I/O** | | Player(int x, int y) | Creates a new player at x,y | None | None | None | | Rectangle getRect() | Returns the rectangle occupied by the player. | None | None | None | | void restoreImg() | Reassigns filepath. | None | None | None. | |

|  |
| --- |
| GameObject:  Holds the player’s character. |
| Instance variables:   |  |  |  | | --- | --- | --- | | **Name** | **Type** | **Description** | | img | Image | Stores thesprite. | | x | int | Holds the coordinates. | | y | |
| Member methods:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Description** | **Pre/post conditions** | **Interaction** | **I/O** | | GameObject(int x, int y) | Creates a new object at x,y | None | None | None | | Rectangle getRect() | Returns the rectangle occupied by the player. | None | None | None | | void restoreImg() | Reassigns filepath. | None | None | None. | | void move() | Holds the movement logic. | None | None |  | | void collide(Player p) | Holds the collision handler. | None | Calls p’s modifiers | None | | Image getImage() | Returns the sprite | None | None | None | | boolean visibleIn() | Determines if the object is visible in ‘r’. | None | None | None | |

|  |  |  |
| --- | --- | --- |
| Subclasses of GameObject:  These classes all extend GameObject and define some sort of thing the player can interact with. | | |
| # | **Name** | **Description** |
| 1 | Obstacle | A basic obstacle that removes a life if you collide into it. |
| 2 | Life | Stationary object that gives the player a life on pickup. |
| 3 | Missile | Flies upward, killing the player if touched. |
| 4 | Sentry | Flies in clockwise squares. Kills the player if touched. |
| 5 | Lightning | Moves left to right. Damages the player on contact. |
| 6 | Astronaut | Drifts in a twisted-8 pattern. Points awarded if rescued. |
| 7 | Black Hole | Moves left to right before teleporting back to original position. Teleports the player if touched. |
| 8 | Scout. | Moves in compass pattern. Kills if touched. |
| 9 | Technology | Drifts around. Awards points and lives on retrieval. |

Manual:

* To move up, down or right, use WSD.
* Try to collect lives, rescue astronauts, and retrieve alien technology.
* Black holes teleport you around- this could be useful or dangerous.
* Avoid everything else.