Marmara University - Faulty of Engineering

Department of Computer Engineering

CSE1242 Computer Programming 2 (Spring 2022)

Submit Date: 02/05/2022

Game (Puzzle)

|  |  |  |
| --- | --- | --- |
| Student Number (ID) | Name | Surname |
| 150120998 | Abdelrahman | Ahmed Mohamed Nabil Zahran |
| 150121534 | Furkan | Acar |

Sections Of the Report: -

* Section (1): Problem Definition + Game Parts.
* Section (2): Implementation Details.
* Section (3): UML Diagrams.
* Section (4): Test Cases.

Section (1): Definition + Game Parts:-

In this term project we will build a (pipe tiles - puzzle) game using java programming language (java fx developing tool). This game aims to move a ball on a game board from a starting point (Starter Tile) to an end point (End Tile) located on the board. The main target of this game is to try to put the piped tiles in a correct order to create a path from the start point to the end point, so the ball can reach the end point and the player proceeds to the next level. This is a one player game, but you can always get help form family and friends, hope you enjoy it.

This game is made of 11 levels starts with level number 1 and ends with level number 11; the player can’t go to the next level without passing the current level. In each level the player must find the correct path so the ball can reach the endpoint and he/she can move to the next level. The game board is constructed of 16 tiles (4 x 4). Each level is a totally different challenge to the player to provide the player with the best experience through the game. The number of tiles is the same through all the levels, there are 7 types of tiles used in the game board. These types are (Start tile – End Tile – Empty Tiles (movable or not movable) – Pipe tiles (movable or not movable) – Curved Pipe Tiles (movable or not movable)). In each level, there is only one start tile and one end tile while the left tiles are from the other types. Each type of tiles has a different color.

Section (2): Implementation

In the implementation of this game, we used the javafx developing tool and its imported libraries provided by java language.

This project is done using 2 different packages: -

* Application package: -
* Helper Class
* EmptyFreeTile Class
* PipeStatic Class
* Repository Class
* Main Class
* CurvedPipe Class
* PipeTile Class
* StartTile Class
* Types Class
* MenuItems Class
* OpenPageTitle Class
* EmptyTile Class
* Properties Class
* CurvedPipeStatic Class
* EndTile Class



* Application.Abstract package: -
* Movable Class
* Pipe Class
* Tile Class



Diagram

Description automatically generated with medium confidence

Section (3): UML Diagrams

* Application package: -

The Application Abstract package contains all the abstract classes of the Game project (puzzle) in addition to the special tile class.

All these classes are imported to the Application package where their abstracted methods are implemented differently for each class in different cases to provide a wide usage of these methods.

|  |  |
| --- | --- |
| <<interface>>  Movable | |
|  |  | |
|  |  | |

* The movable interface is used to show the adjective movement of the tiles – implemented in different classes later to show the tiles’ movements.

|  |  |
| --- | --- |
| *Tile* | |
| **-**  **+**  **+**  **-**  **-**  **+**  **-** | id: String  typeNormal: String  propertyNormal: String  type: Types  property: Properties  path: String  image: Image | |
| **+**  **+**  **+**  **+**  **+**  **+**  **+**  **+**  **+**  **+**  **+**  **+**  **+**  **+**  **+** | Tile (tile: Tile)  Tile (id: String, type: String, property: String, path: String)  equals (obj: Object): Boolean  getImageFromVisual (): Image  setImage (path: String): void  getTileId (): String  setTileId (id: String): void  getTypes (): Types  setTypes (type: String): void  goToRight (): void  goToLeft (): void  goToUp (): void  goToDown (): void  getPropertiesFromTile (): Properties  setProperties (property: String): void | |

* The Tile Class is a subclass of the Pane class. This class is created to apply special effects from the javafx libraries on the menu items (buttons) of the open page of the game.
* Data Field: id of the String type (tile id).
* Data Field: typeNormal of the String type (tile type).
* Data Field: propertyNormal of the String type (tile property).
* Data Field: type of the types type.
* Data Field: property of the properties type.
* Data Field: path of the String type (image type).
* Data Field: image of the Image type.
* The Tile() constructor is used to create tile(Movable or non-Movable) objects used to control the tile on the game board and defines its position + direction using (id, type, property, path(image)) then add that object to the path(image list) to display the tile on the screen.
* The second Tile() constructor is used to create an object of the tile with the type of class tile which is used to control movement properties of the tile and the position of it.
* The equals() method is used to check the objects of the tiles.
* The getImageFromVisual() method is the Getter method of image of the tile.
* The setImage() method is the Setter method of image the of the tile.
* The getTileId() method is the Getter method of id of the tile.
* The setTileId() method is the Setter method of id the of the tile.
* The getTypes() method is the Setter method of type of the tile.
* The setTypes() method is the Setter method of type of the tile.
* The goToRight() method move the tile to the right.
* The goToLeft() method move the tile to the left.
* The goToUp() method move the tile up.
* The goToDown() method move the tile down.
* The getPropertiesFromTile() method is the Getter method of property of the tile.
* The getPropertiesFromTile() method is the Setter method of property of the tile.
* The imported libraries and classes used in the Tile Class: -
* application.Properties
* application.Types
* javafx.animation.TranslateTransition
* javafx.scene.image.Image
* javafx.scene.image.ImageView

|  |  |
| --- | --- |
| *pipe* | |
|  |  | |
| **+**  **+**  **+**  **+** | Pipe (pipe: Pipe)  Pipe (id: String, type: String, property: String, path: String)  *addPath (): void*  *isContinue (twoDim: Tile [][], previousTile: Tile): boolean* | |

* The Pipe Class is a subclass of the Tile Class. The Pipe Class is an abstract class. This class is constructed using 2 constructors and 2 methods. Each pipe Tile has a path.
* The Pipe() constructor is used to create pipe tile(Movable or non-Movable) objects used to control the pipe tiles on the game board and define their positions + directions using (id, type, property, path(image)) then add these objects to the path(image list) to display the pipe tile on the screen.
* The second Pipe() constructor is used to create objects of the pipe tiles with the type of class pipe which is used to control movement properties of the tile and the position of it.
* The addPath() method is an abstract method.

is used later in the other classes to add new path after the tile + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.

* The isContinue() method is an abstract method.

is used later in the other classes to add new pipe path for the ball on the game board + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.

* Application.Abstract package: -

|  |  |
| --- | --- |
| Main | |
| +  +  +  +  +  +  -  -  -  -  -  - | objectForMove: Tile  pathTransition: PathTransition  duration: final int  numberMovement: int  isCompleted: boolean  primaryStageGlobal: Stage  WIDTH: final int  HEIGHT: final int  root: Pane  menuBox: VBox  line: Line  menuData: List<Pair<String, Runnable>> | |
| **+**  **+**  **-**  **-**  **-**  **+**  **+**  **-**  **+**  **-**  **-**  **-**  **-**  **-**  **-** | generateGameBoard(primaryStage: Stage): void  start(primaryStage: Stage): void  setScene(primaryStage: Stage, root: BorderPane): void  moveTiles(tileList: ArrayList<Tile>, twoDim: Tile[][],tile: Tile, levelName: Label): void  checkIsGameDone(tileList: ArrayList<Tile>, twoDim: Tile[][]): void  createBall(): Circle  changePositionOfTwoObjectInList(list: Tile[][],to: Tile, from: Tile): void  setBackgroundImage(root: BorderPane): void  main(args: String[]): void  createContent(): Parent  addBackground(): void  addTitle(): void  addLine(x: double, y: double): void  startAnimation(): void  addMenu(x: double, y: double): void | |

* The StartTile Class is a subclass of the Pipe Class. This class is used to define . This class is constructed using 15 methods.
* The generateGameBoard() - Create the the game board by generating the tiles on the screen(Stage) + 2 Horizontal Boxes for 1 - (Level Name + Number of Movements) 2 - (next level + previous Button).
* The Start() method - display the stage on the screen for the user.
* The
* The moveTiles() method - Movement of the tiles of the Game Board.
* The checkIsGameDone() - Check if the Current level is completed + The ball animation executed.
* The createBall() Create Ball of the game board using a circle object.
* The changePositionOfTwoObjectInList() - (movement) exchanging the positions of the objects generated by these tiles.
* The setBackgroundImage() - display the background of the game board.
* The Launch() method - Start Execution + Launch the program.
* The createContent() - Launching the animation and the other methods invoked in this method.
* The addBackGround() - Add Background to the pane - Display.
* The addTitle() method - add the title of the open page to the pane - Display.
* The addLine() - create the line object
* The startAnimation() - start the animation for the menu of the open page.
* The addMenu() is to create the menu box and then add it to the pane to display it on the screen.

|  |  |
| --- | --- |
| MenuItems | |
| **-**  **-**  **-** | text: Text  shadow: Effect  blur: Effect | |
| **+**  **+** | MenuItems (name: String)  setOnAction (action: Runnable): void | |

* The MenuItems Class is a subclass of the Pane class. This class is created to apply special effects from the javafx libraries on the menu items (buttons) of the open page of the game.
* Data Field: text of the class Text type (subclass of scene in javafx) used to create text objects that displays a text on the screen of the menu items (Buttons).
* Data Field: shadow of the class Effect type (subclass of scene in javafx) used to create shadow object which is used to apply a shadow effect on the menu items.
* Data Field: blur of the class Effect type (subclass of scene in javafx) used to create blur object which is used to apply a blur effect on the menu items.
* The MenuItems() method is used to Create the items of the menu + modifying them and applying requested fonts + colors + Effects and etc..
* The setOnAction() is used to create a mouse event (action) in order to make the mouse actions applicable.
* The hoverProperty() of javafx is used to determine the activation time of special effects - Whether or not this Node is being hovered over. Typically, this is due to the mouse being over the node, though it could be due to a pen hovering on a graphics tablet or other form of input.

|  |  |
| --- | --- |
| OpenPageTitle | |
| **-** | text: Text | |
| **+**  **+**  **+** | OpenPageTitle(name: String)  getTitleWidth(): double  getTitleHeight(): double | |

* The OpenPageTitle Class is a subclass of the Pane class. This class is created to apply special effects from the javafx libraries on title of the open page of the game.
* Data Field: text of the class Text type (subclass of scene in javafx) used to create a text object that displays a text on the screen of the title.
* The OpenPageTitle() method is used to set the title of the open page properties + fonts.
* The getTitleWidth() is used to get the width of the layout bounds of the node.
* The getTitleHeight() is used to get the height of the layout bounds of the node.

There is only one starter tile in a game board. Starter tiles cannot move; their locations are static. Starter tiles may have horizontal or vertical pipes inside.

|  |  |
| --- | --- |
| StartTile | |
|  |  | |
| **+**  **+**  **+**  **+** | StartTile (id: String, type: String, property: String, path: String)  StartTile (tile: StartTile)  addPath (): void  isContinue (twoDim: Tile [][], previousTile: Tile): boolean | |

* The StartTile Class is a subclass of the Pipe Class. This class is used to define the start tile (non-Movable) on the game board. This class is constructed using 2 constructors and 2 methods. Each Start Tile has a position on the game board (id) and direction (property).
* The StartTile() constructor is used to create start tile(non-Movable) objects used to control the start tile on the game board and defines its position + direction using (id, type, property, path(image)) then add that object to the path(image list) to display the start tile on the screen.
* The second StartTile() constructor is used to create an object of the start tile with the type of class tile which is used to control movement properties of the tile and the position of it.
* The addPath() method is used to add new path after the tile + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The isContinue() method is used to add new pipe path for the ball on the game board + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The LineTo feature of javafx is used to create an instance of LineTo to create lines of the pipes using endpoint coordinates where (x is the horizontal coordinate of the line end point) + (y is the vertical coordinate of the line end point.)
* The MoveTo feature of javafx is used to create an instance of MoveTo to create a movement of the pipes using endpoint coordinates where (x is the horizontal coordinate of the line end point) + (y is the vertical coordinate of the line end point.)

|  |  |
| --- | --- |
| Static pipe property is specified as “**PipeStatic**” in input file. Static pipes may be horizontal or vertical. **PipeStatic** tiles cannot move; their locations are static. | |
| PipeStatic | |
|  |  | |
| **+**  **+**  **+**  **+** | PipeStatic (id: String, type: String, property: String, path: String)  addPath (): void  findTileInTwoDim (list: Tile [][], tile: Tile): int []  isContinue (twoDim: Tile [][], previousTile: Tile): boolean | |

* The PipeStatic Class is a subclass of the Pipe Class. This class is used to define the pipe tiles (non-Movable) on the game board. This class is constructed using one constructor and 3 methods. Each Pipe Tile has a position on the game board (id) and direction (property).
* The PipeStatic() constructor is used to create pipe tiles(non-Movable) objects’ used to control the pipe tiles on the game board and defines their positions + directions using (id, type, property, path(image)) then add these objects to the path(image list) to display these tiles on the screen.
* The addPath() method is used to add new path after the tile + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The isContinue() method is used to add new pipe path for the ball on the game board + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The findTileInTwoDim() method is used to define the position of the selected tile using (x, y) coordinates system as a (Point) using a search algorithm and a n equality control check (Checking the equality of the tile id till the same id one is found (point 1 == point 2)).

|  |  |
| --- | --- |
| There may be many pipes in a game board. Pipes may be horizontal or vertical. Pipes may move to empty free tiles. | |
| PipeTile | |
|  |  | |
| **+**  **+**  **+** | PipeTile (id: String, type: String, property: String, path: String)  addPath (): void  isContinue (twoDim: Tile [][], previousTile: Tile): boolean | |

* The PipeTile Class is a subclass of the Pipe Class and implements the Movable interface. This class is used to define the pipe tiles (Movable) on the game board. This class is constructed using one constructor and 2 methods. Each Pipe Tile has a position on the game board (id) and direction (property).
* The PipeTile() constructor is used to create pipe tiles(Movable) objects’ used to control the pipe tiles on the game board and defines their positions + directions using (id, type, property, path(image)) then add these objects to the path(image list) to display these tiles on the screen.
* The addPath() method is used to add new path after the tile + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The isContinue() method is used to add new pipe path for the ball on the game board + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The LineTo feature of javafx is used to create an instance of LineTo to create lines of the pipes using endpoint coordinates where (x is the horizontal coordinate of the line end point) + (y is the vertical coordinate of the line end point.)
* The MoveTo feature of javafx is used to create an instance of MoveTo to create a movement of the pipes using endpoint coordinates where (x is the horizontal coordinate of the line end point) + (y is the vertical coordinate of the line end point.)

|  |  |
| --- | --- |
| Empty tiles are tiles without pipe. There are two types of empty tiles. One presented in Figure 6 stands for an empty space. Other movable tiles can move to the location of this empty tile. Their property is specified as “free” in input file. | |
| EmptyFreeTile | |
|  |  | |
| **+** | EmptyFreeTile (id: String, type: String, property: String, path: String) | |

* The EmptyFreeTile Class is a subclass of the Tile Class and implements the Movable interface. This class is used to define the Empty tiles (Movable) on the game board. This class is constructed using one constructor. Each Empty Tile has a position on the game board (id) and direction (property).
* The EmptyFreeTile() constructor is used to create pipe tiles objects’ used to control the pipe tiles on the game board and defines their positions + directions using (id, type, property, path(image)) then add these objects to the path(image list) to display these tiles on the screen.

|  |  |
| --- | --- |
| The empty tile presented in Figure 7 stands for an empty space. Tiles cannot move to the location of this empty tile. To give an example; tile 10 in Figure 3 cannot move to tile 11. However, tile 10 can move to the place of tile number 14. Their property is specified as “**none**” in input file. Empty tiles may move to empty free tiles. | |
| EmptyTile | |
|  |  | |
| **+**  **+** | EmptyTile (id: String, type: String, property: String, path: String)  EmptyTile (tile: EmptyTile) | |

* The EmptyTile Class is a subclass of the Tile Class. This class is used to define the Empty tiles (non-Movable) on the game board. This class is constructed using 2 constructors. Each Empty Tile has a position on the game board (id) and direction (property).
* The EmptyTile() constructor is used to create Empty tiles objects’ used to control the Empty tiles on the game board and define their positions + directions using (id, type, property, path(image)) then add that object to the path(image list) to display the Empty tiles on the screen.
* The second EmptyTile() constructor is used to create objects of the Empty tiles with the type of class tile which are used to control movement properties of the tile and the position of it.

|  |  |
| --- | --- |
| Curved pipes may move to empty free tiles. The type of a curved pipe is represented as “**Pipe**” in input file. Curved pipe property may take values of “**00, 01, 10, 11**”, each representing the curve type. There may be four different curved pipe alignments. The exact representation of each alignment is illustrated in Figure 10. If the property value of a tile is equal to **00**, then it corresponds to the top left tile in Figure 10. As an example, pipe type is **01** is used for the tile number 13 in the sample game board given in Figure 3. It should be noted that, the numbers (**00, 01, 10, 11**) are just used for curve type representations, they do not represent tile IDs. | |
| CurvedPipe | |
|  |  | |
| **+**  **+**  **+**  **+** | CurvedPipe (id: String, type: String, property: String, path: String)  addPath (): void  findTileInTwoDim (list: Tile [][], tile: Tile): int []  isContinue (twoDim: Tile [][], previousTile: Tile): boolean | |

* The CurvedPipe Class is a subclass of the Pipe Class and implements the Movable interface. This class is used to define the curved pipe tiles (Movable) on the game board. This class is constructed using one constructor and 3 methods. Each Curved Pipe Tile has a position on the game board (id) and direction (property).
* The CurvedPipe() constructor is used to create pipe tiles(Movable) objects’ used to control the pipe tiles on the game board and defines their positions + directions using (id, type, property, path(image)) then add these objects to the path(image list) to display these tiles on the screen.
* The addPath() method is used to add new path after the tile + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The isContinue() method is used to add new pipe path for the ball on the game board + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The findTileInTwoDim() method is used to define the position of the selected tile using (x, y) coordinates system as a (Point) using a search algorithm and a n equality control check (Checking the equality of the tile id till the same id one is found (point 1 == point 2)).
* The CubicCurveTo feature of javafx is used to Create a curved path element, defined by three new points,by drawing a Cubic Bézier curve that intersects both the current coordinates and the specified coordinates (x,y) using the specified points (control X1,control Y1) and (controlX2,controlY2) as Bézier control points. All coordinates are specified in double precision.

|  |  |
| --- | --- |
| Curved pipes may move to empty free tiles. The type of a curved pipe is represented as “**Pipe**” in input file. Curved pipe property may take values of “**00, 01, 10, 11**”, each representing the curve type. There may be four different curved pipe alignments. The exact representation of each alignment is illustrated in Figure 10. If the property value of a tile is equal to **00**, then it corresponds to the top left tile in Figure 10. As an example, pipe type is **01** is used for the tile number 13 in the sample game board given in Figure 3. It should be noted that, the numbers (**00, 01, 10, 11**) are just used for curve type representations, they do not represent tile IDs. | |
| CurvedPipeStatic | |
|  |  | |
| **+**  **+**  **+**  **+**  **+** | CurvedPipeStatic (pipe: Pipe)  CurvedPipeStatic (id: String, type: String, property: String, path: String)  addPath (): void  findTileInTwoDim (list: Tile [][], tile: Tile): int []  isContinue (twoDim: Tile [][], previousTile: Tile): boolean | |

* The CurvedPipe Class is a subclass of the Pipe Class. This class is used to define the curved pipe tiles (non-Movable) on the game board. This class is constructed using 2 constructors and 3 methods. Each Curved Pipe Tile has a position on the game board (id) and direction (property).
* The CurvedPipe() constructor is used to create curved pipe tiles (non-Movable) objects’ used to control the curved pipe tiles on the game board and defines their positions + directions using (id, type, property, path(image)) then add these objects to the path(image list) to display these tiles on the screen.
* The second CurvedPipe() constructor is used to create an object of the curved pipe tile with the type of class pipe which is used to control movement properties of the tile and the position of it.
* The addPath() method is used to add new path after the tile + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The isContinue() method is used to add new pipe path for the ball on the game board + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The findTileInTwoDim() method is used to define the position of the selected tile using (x, y) coordinates system as a (Point) using a search algorithm and a n equality control check (Checking the equality of the tile id till the same id one is found (point 1 == point 2)).
* The CubicCurveTo feature of javafx is used to Create a curved path element, defined by three new points,by drawing a Cubic Bézier curve that intersects both the current coordinates and the specified coordinates (x,y) using the specified points (control X1,control Y1) and (control X2,control Y2) as Bézier control points. All coordinates are specified in double precision.

|  |  |
| --- | --- |
| There is only one end tile in a game board. End tiles cannot move; their locations are static. End tiles may have horizontal or vertical pipes inside. | |
| EndTile | |
|  |  | |
| **+**  **+**  **+ +** | EndTile (id: String, type: String, property: String, path: String)  addPath (): void  findTileInTwoDim (list: Tile [][], tile: Tile): int []  isContinue (twoDim: Tile [][], previousTile: Tile): boolean | |

* The EndTile Class is a subclass of the Pipe Class. This class is used to define the End tile (non-Movable) on the game board. This class is constructed using one constructor and 3 methods. Each Start Tile has a position on the game board (id) and direction (property).
* The EndTile() constructor is used to create End tile (non-Movable) objects used to control the End tile on the game board and defines its position + direction using (id, type, property, path(image)) then add that object to the path(image list) to display the end tile on the screen.
* The addPath() method is used to add new path after the tile + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The isContinue() method is used to add new pipe path for the ball on the game board + Check the surrounding tiles (Horizontally + Vertically) one by one their types and either they are movable or non-movable.
* The findTileInTwoDim() method is used to define the position of the selected tile using (x, y) coordinates system as a (Point) using a search algorithm and a n equality control check (Checking the equality of the tile id till the same id one is found (point 1 == point 2)).
* The LineTo feature of javafx is used to create an instance of LineTo to create lines of the pipes using endpoint coordinates where (x is the horizontal coordinate of the line end point) + (y is the vertical coordinate of the line end point.)

|  |  |
| --- | --- |
| Helper | |
| **+** | contents: String[] | |
| **+**  **+**  **+** | readFromFile(tileList: ArrayList<Tile>): void  generateTiles(list: String[]): Tile  generateTilesFromTile(tile: Tile): Tile | |

* The EndTile Class is a subclass of the Pipe Class. This class is used to define the End tile (non-Movable) on the game board. This class is constructed using one constructor and 3 methods. Each Start Tile has a position on the game board (id) and direction (property).
* Data Field: contents of the type String[] is used to store the file contents.
* The static method readFromFile() is used to read input files and take from them to use it to construct the game board for each level.
* The generateTiles() method generates tiles is used for creating objects of the tiles.
* The method Tile generateTilesFromTile() generates tiles is used to create objects in order to change the reference type of the objects of the tiles which were generated by the generateTiles() method.

|  |  |
| --- | --- |
| Repository | |
| **+**  **+**  **+**  **+**  **+** | pipeList: ArrayList<Pipe>  path: Path  twoDim: Tile[][]  tileList: ArrayList<Tile>  currentLevel: int | |
| **+**  **+**  **+**  **+**  **+** | incrementLevel(): void  decrementLevel(): void  runAgain(): void  findTileInTwoDim(list: Tile[][],tile: Tile): int[]  findXYCoordinate(tile: Tile): int[] | |

* The Repository Class is a class which is created to provide a back up list of the most used data fields (main lists for storing and using for other classes) in the other classes to provide coherence and cohesion to the code and afford a better readability.
* Data Field: pipelist ArrayList of Pipe class type used to store the pipe tiles
* Data Field: path of Path class type used to store the path pipe tiles.
* Data Field: twoDim a 2-dimensional array used to store all the tiles to create the game board.
* Data Field: tilelist ArrayList of Tile class type used to store the tiles and their order to make the movement.
* Data Field: currentLevel of int type which displays the level’s number.
* The incrementLevel() is used to Increase the level number then activate the next level button (move from the current level to the next level if there is).
* The decrementLevel() method is used to decrease the level number then activate the previous level button (move from the current level to the previous level if there is).
* The Static method RunAgain() -> Clear all lists when the current level is completed to update the next level.
* The static method find TileInTwoDim is used to the position of the selected tile.
* The static method findXYCoordinates() of int[] type is used to convert (normal coordinates to pixel coordinates).

|  |  |
| --- | --- |
| Types | |
| **+** | Enum {STARTER, END, EMPTY, PIPE, PIPESTATIC,} | |
|  |  | |

|  |  |
| --- | --- |
| Properties | |
| + | Enum {HORIZONTAL, VERTICAL, NONE, FREE, CURVED\_ZERO\_ZERO, CURVED\_ZERO\_ONE, CURVED\_ONE\_ZERO, CURVED\_ONE\_ONE,} | |
|  |  | |

Section (4): Test Cases

The test cases of all levels screenshots have been attached in the project main zip file.