Design & Implementation:

Please see the UML diagram at the last page of the report for clear view on the implementation of all the interfaces & classes. All the public and private data members has been shown the the diagram.

In Main class, a new Game class object is created. Game object reads the gameGrid.txt & loads it into memory. By loading Game object creates a 2D array of Jewel Class & fills the corresponding Jewel object type in this array.

Further in main, leaderboard.txt is read into an Player ArrayList to be used later. Then user is prompted to play the game & further inputs.

Once user inputs the coordinates, it is checked for validation. If valid, Jewel is fetched for the given coordinates and a match is searched in horizontal, vertical, diagonal or more of these directions depending on the jewel type. If there is a match, an ArrayList of Coordinates for matched jewels are returned. Then Match is applied, means all the matched jewels are replaced with an Empty jewel type and Gravity is applied to the game board. Points are calculated as well.

This continues until user chooses to end the game. In this stage, users point & ranks are displayed on screen and saved in leaderborad.txt file.

Different types of Jewels:

An abstract Jewel class is created which defines the properties of a jewel & define its properties such as coordinates, name, names of other matching jewels, where to look for a match.

JewelMatch is an interface which declares the findMatch() method which is to find the match for different types of jewels.

An abstract class JewelMatchDecorator is creates which implements the JewelMatch class. Then 4 classes: HorizontalMatch, VerticalMatch, RightDiagonalMatch & LeftDiagonalMatch is defined these all classes extends to JewelMatchDecorator. Each of these classes actually defines the findMatch() method to look for a match in horizontal, vertical or diagonal directions & returns an ArrayList of coordinates of matched jewels if the match is found.

All the Jewel types extend to the Jewel class & have a component JewelMatch to perform the corresponding type of match depending on the type of Jewel.

Data Structures used:

ArrayList has been used to store the different types of objects.

Interfaces: Following interfaces has been defined.

JewelMatch

Abstract Classes: Following classes has been defined:
Jewel

JewelMatchDecorator

Classes: Following classes has been defined:

Main

Player

Game

Coordinate

HorizontalMatch

VerticalMatch

RightDiagonalMatch

LeftDiagonalMatch

Plus

Minus

Pipe

Slash

Backslash

Square

Diamond

Triangle

Wildcard

Empty

Steps to add a new Jewel type

Following steps are required:

1. Create a class for new jewel. Let's call it NewJewel:

```
public class NewJewel {

}
```

2. Extend the class to Jewel class: it will require us to implement the findMatch function. Define the function as shown below:

```
public class NewJewel extends Jewel {


	@Override
	public ArrayList<Coordinate> findMatch(Jewel[][] gameGrid) {
		return jewelMatch.findMatch(this, gameGrid);
	}
}
```

3. Define the constructor:
   Constructor should take 2 arguments: x & y coordinates of the jewel.

```
public NewJewel(int x, int y){

	}
```

4. Super class Jewel, accepts 3 arguments:

```
super("N", 30, new Coordinate(x,y));
```

   1st : Name of NewJewel
   2nd : Points for the NewJewel
   3rd: Coordinate object for x & y.

5. Define the match rule for NewJewel in constructor:

```
this.jewelMatch = new HorizontalMatch(new RightDiagonalMatch(new
LeftDiagonalMatch()));
```

   Above defines, a rule to match first left diagonal then right diagonal then match horizontally.
   Similarly,

```
Rule: this.jewelMatch = new VerticalMatch(new HorizontalMatch(new
RightDiagonalMatch()));
```

   Defines to match first right diagonal then horizontally then vertically.

6. In constructor, Add Matching jewel type names:
   Add the names of the all the jewels which will match this jewel type.

```
addMatchingJewelNames(this.name);
addMatchingJewelNames("D");
addMatchingJewelNames("S");
```

   For example in above, This NewJewel will match with "N" : NewJewel, "D" : Diamond and "S" : Square

7. In Game.java, add an “if else” for this type in **`loadGameGridFromFile()`** method:

```
else if (jewelName.equals("N")){
      jewel = new NewJewel(i,j);
}
```

And the new Jewel type is ready to be used. It’s very simple.

UML Class Diagram