2. Software Using Documentation

2.1. Software Usage

Program will NOT take gameGrid.txt and leaderboard.txt as command arguments. It will simply assume that those file names are fixed and are present in program dictionary..

2.2 Error Messages

Only error message for user is when s/he selects an empty space as a coordinate where program prints "Selected Jewel is already deleted, please select another one"

3. Software Design Notes

3.1. Description of the program

3.1.1. Problem

The problem is to develop a game which will behave like Candy Crush or Bejeweled, where user destroys the triplet of jewels upon meeting that jewel's condition.

3.1.2. Solution

Solution should be object oriented and flexible. If the conditions change and/or another Jewel type with different set of rules are desired, system should support that kind of maintainability and flexibility.

3.2. Main Data Structures

Game grid is defined as 2D array whose dimensions are determined during run time. This grid can be of any size MxN. As for another type Jewels will store their position(x,y), name as a single character ('D','+','W'..) and their point. In addition each will have a check function which will be implemented for every one of them and check for different conditions. Last but not least for leader board, each entry is stored as an object and all of those objects are hold with an ArrayList<Score> collection.

3.3. Algorithm

```
1. Make initialization.
    1.1. Read gameGrid.txt and learn dimensions and create a game grid object with them.
    1.2. For each char in gameGrid.txt
        1.2.1 Determine the type of that particular jewel
        1.2.2 Create it and put it to board
        1.2.3 Increment the coordinate by one (take mod if reached end of row)

2. Take commands from user
    2.1 Read a coordinate from user and check if it is empty.
    2.2 If the coordinate contains a jewel invoke its check function which will check for
required conditions for current jewel and move to another upon meeting those.
    2.3 When a triplet found, delete them from grid, award user with points.
    2.4 Before asking for another input make jewels fall (when there is an empty space).
    2.5 Repeat this until user types 'E'.

3. Update Leader board
   3.1 Turn all existing scores into a score object array.
   3.2 If this score is not already in this collection add it, sort the collection and perform a
binary search for it and it's closest neighbors.
   3.3 Print players place in leader board and finish the program by typing "Good bye"
```

4. Special Design Issues

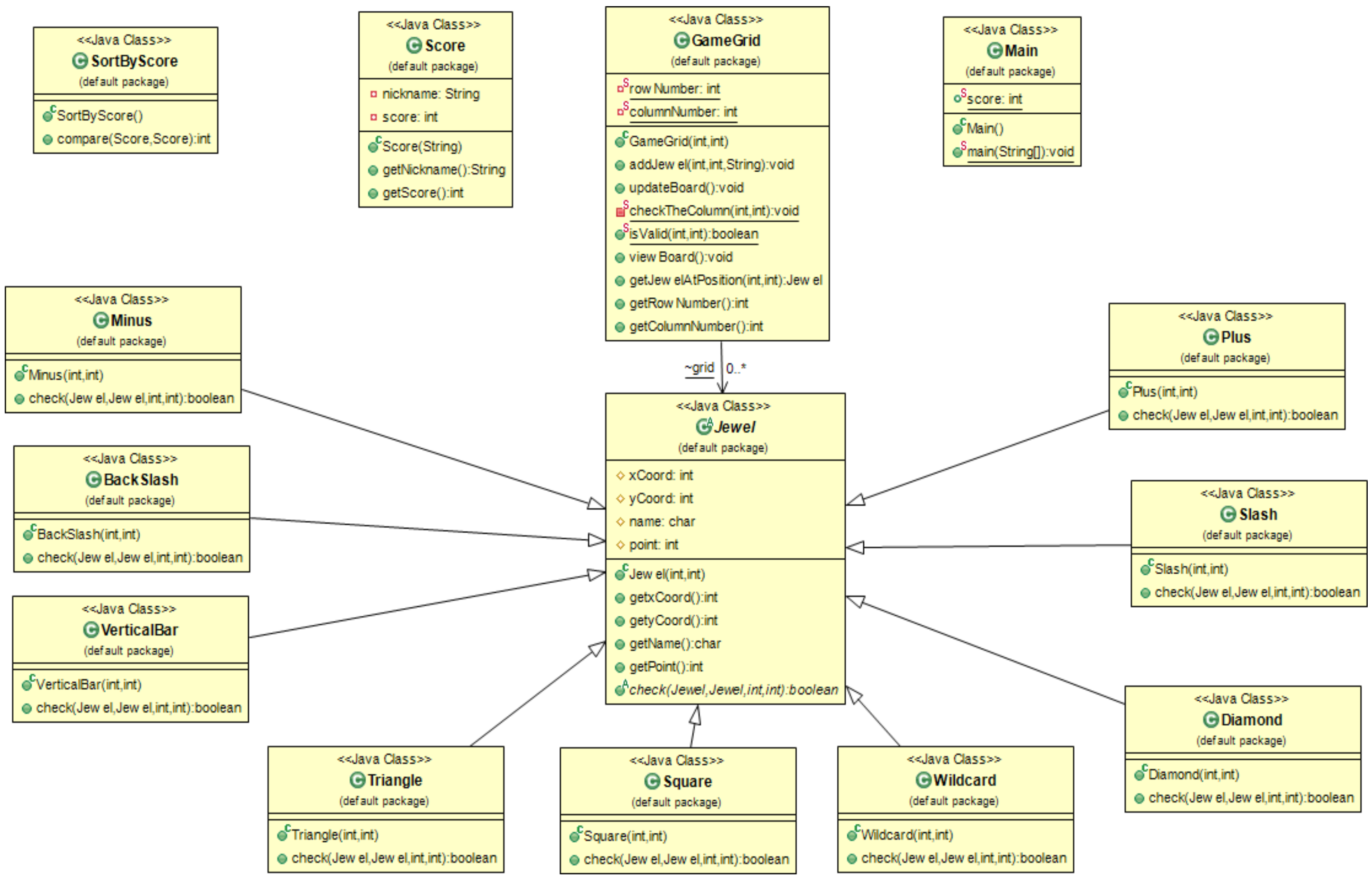
4.1 Adding A New Jewel

Adding a new jewel type with a new rule set will not require a huge effort since all Jewels are inherited from a single abstract class, their methods and attributes are predefined. Only the corresponding check() function will require an implementation which will hold that jewel's triplet matching conditions. In addition, addJewel() method in GameGrid class need to be updated in such a way to support new jewel's constructor. A simple if clouse will be enough for that.

Other than that, the remaining parts of the code can stay as it is and will work like a charm with existing jewels.

4.2 Creating Random Jewels For Empty Spaces

That should also be fairly simple, currently empty spaces are indicated by null pointer. Instead of doing that programmer can assign those empty spaces to a randomly created new jewel. That way after user makes a move, deleted parts will be filled with new jewels. Of course that could lead the game to state where player can not make a valid move and forced to end the game which is not ideal.



<<Java Class>> SortByScore (default package)
SortByScore()
compare(Score,Score):int
<<Java Class>> Score (default package)
nickname: String
score: int
Score(String)
getNickname():String
getScore():int
<<Java Class>> GameGrid (default package)
rowNumber: int
columnNumber: int
GameGrid(int,int)
addJewel(int,int,String):void
updateBoard():void
checkTheColumn(int,int):void
isValid(int,int):boolean
viewBoard():void
getJewelAtPosition(int,int):Jewel
getRowNumber():int
getColumnNumber():int
<<Java Class>> Main (default package)
score: int
Main()
main(String[]):void
~grid
0..*
<<Java Class>> Jewel (default package)
xCoord: int
yCoord: int
name: char
point: int
Jewel(int,int)
getxCoord():int
getyCoord():int
getName():char
getPoint():int
check(Jewel,Jewel,int,int):boolean
<<Java Class>> Minus (default package)
Minus(int,int)
check(Jewel,Jewel,int,int):boolean
<<Java Class>> BackSlash (default package)
BackSlash(int,int)
check(Jewel,Jewel,int,int):boolean
<<Java Class>> VerticalBar (default package)
VerticalBar(int,int)
check(Jewel,Jewel,int,int):boolean
<<Java Class>> Plus (default package)
Plus(int,int)
check(Jewel,Jewel,int,int):boolean
<<Java Class>> Slash (default package)
Slash(int,int)
check(Jewel,Jewel,int,int):boolean
<<Java Class>> Diamond (default package)
Diamond(int,int)
check(Jewel,Jewel,int,int):boolean
<<Java Class>> Triangle (default package)
Triangle(int,int)
check(Jewel,Jewel,int,int):boolean
<<Java Class>> Square (default package)
Square(int,int)
check(Jewel,Jewel,int,int):boolean
<<Java Class>> Wildcard (default package)
Wildcard(int,int)
check(Jewel,Jewel,int,int):boolean