HACETTEPE UNIVERSITY DEPARTMENT OF COMPUTER ENGINEERING
BBM104

Prepared by Burak YILMAZ
Number: 21627868
E-Mail: burak040898@gmail.com
Subject: Assignment 4

MAIN GOAL

In this assignment, our main goal is using interfaces or abstract classes or both and recreate a simplified version of the game called "Bejeweled"

Blue color represents classes.

Green color represents method names and type of method.

Purple color represents txt names.

Orange color represents keywords.

PROBLEM

In this problem, the game consists of a grid (of any size, e.g. 10x10) of “jewels”. Some example jewels are Diamond (D), Square (S), Triangle (T) and Wildcard (W). The goal is to find three jewels that are match in a row, column or diagonal by selecting the right coordinate. When this occurs, the three jewels are deleted, and other jewels fall from the top to fill in gaps. If the selected coordinate does not have any triple to match, then a warning message is displayed to the user and a new coordinate is requested. Each jewel match in a different way. Each jewel has got own specific points and the coordinate specified by user can be either first or last item in triple. The triplet should be searched using the rules. Also, mathematical symbol jewels as well. All Mathematical symbols are worth 20 points. Our program should read the gameGrid.txt file for the initial grid and jewels. You should use abstract class or interfaces to implement common characteristics of the jewels.

ALGORITHM

According to my algorithm I created an abstract class called "Bejeweled". It contains some abstract methods and also not abstract methods. Then I created two different abstract classes called "Jewels" and "Math symbols" the idea of this type of a math symbols can match with all other math symbols so that while I was using "instance of" keyword it makes my job easier. After that I began to create each jewel type (jewel type classes = Wildcard, Square, Diamond, Triangle) or math symbol type (math symbol type classes = BackSlash, Slash, Pipe, Plus, Minus). I put an abstract method to "Bejeweled" class called "Bejeweled Detector" and each type of math symbols and jewels override that method so that they have their own match rules. "Wildcard" looks every direction because of that writing the wildcard method means writing the other types method. When I finished "Wildcard" method I finish all types methods. I moved on with creating a class called "Players" for storing each player who played the game. For reading commands I created a class called "Commands" and a created a class called "Main" for executing the whole code. After creating classes I read the gameGrid and store in an array list which each index of array list store map. When user wrote the coordinates if it matches correctly I shifted the jewels correctly again. While user is writing valid coordinates, it moves on like this if user writes an invalid coordinate it requests for another coordinate. If user write E, it ends the program after writing the rank of current player and score. With this hierarchy, it is easy to add a new jewel or math symbol. For example, adding a rectangle jewel all I need to do is create a class called "Rectangle" than it extends "Jewel" class and choose correct rules from Wildcard then it is finished.

Classes

Bejeweled Class: This is an abstract class it has got seven methods. This class takes place at the top of the hierarchy.

Jewel, MathSymbols Classes: These classes extends "Bejeweled" and they are also abstract class so they don’t need to override abstract methods of Bejeweled class.

Wildcard, Square, Triangle, Diamond Classes: These classes extends the "Jewel" class so by extending that class they override "Bejeweled Detector" method (Bejeweled class has got) to create their own matching and looking rules and they all have a variable called point.

Minus, Plus, BackSlash, Slash, Pipe Classes: These classes extends "Math Symbols" class so by extending that they override "Bejeweled Detector" method ("Bejeweled" class has got) to create their own matching and looking rules and they, all have a variable called point.

Command Class: This class reads the testcase.txt and decided which methods are going to use or which variable are going to create.

Main Class: It runs the whole code.

Players Class: This class keeps every different player. It has got several private variables and get methods and it overrides equals method and hash code. Also, it implements comparable because I must put all players to order by looking their points.

Methods

public List<Map> BejeweledDetector (List<Map> jeweled_list,int row,int column,Integer x_axis,Integer y_axis,List<Bejeweled> scorer, Bejeweled target) : This method overrides each type of math symbol and jewel. It looks the tiered order which was given us in Pdf and if it matches it deletes triple.

public abstract List<BeJeweled> Score(List<Bejeweled> scorer): This method overrides in "Jewel" class and "Math symbol" class. It returns only a list.

public abstract Integer getPoint():This method overrides each type of math symbol and jewel.It returns the private variable point.

public void Grid_writer(List<Map> jeweledList,int column,int row):This method writes the grid to the screen. If it encounters with blank it continues without an error. It writes the grid correctly. It takes place under "Bejeweled" class.

public void Grid_Shifter(List<Map> jeweledList,int row,int column):This method shifting the grid correctly and it takes place under "Bejeweled" class.

public int Score_Writer(List<BeJeweled> scorer): This method writes the score that each level not the whole score. It takes place under "Bejeweled" class.

public void leaderboard_writer(List<Players> players, File file): This method writes the leaderboard.txt. If it encounters with same name it writes the maximum value for that name. It takes place under "Bejeweled" class.

public boolean equals (Object o): It is an override method. It returns true if it equals the given object. It takes place under "Players" class.

public int hashCode():It is an override method. It takes place under "Players" class.

public boolean equals_detector(List<Players> players, Players current_player):This method controlling the current player. If point and name is same player could not add to the players list. It takes place under "Players" class.

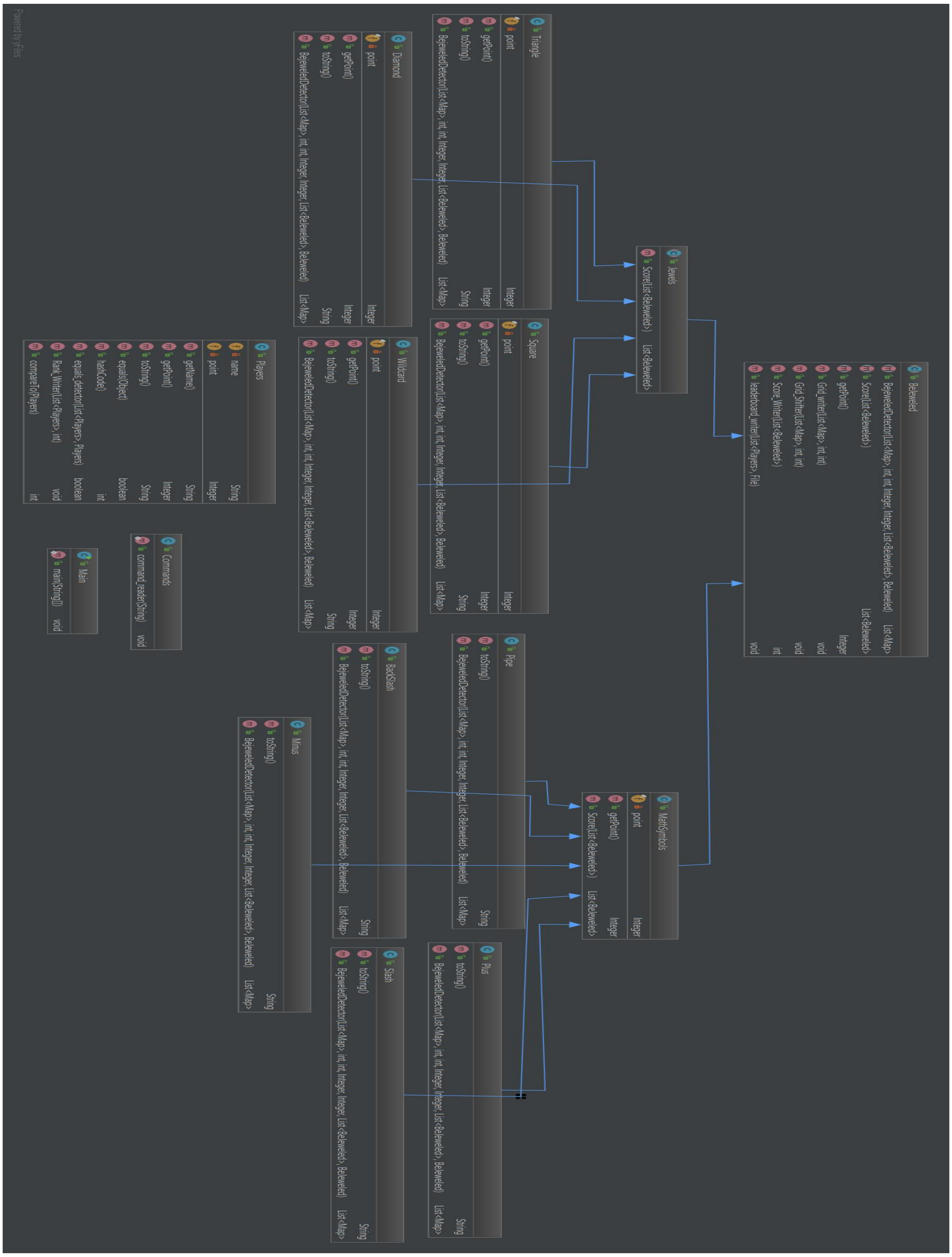
public int compareTo(Players o):It compares the players and also an override method. It takes place under "Players" class.

public void Rank_Writer(List<Players> players, int index): This method writes to the screen. It calculates the rank of the whole player by looking the whole players who played the game before. It takes place under "Players" class.

Algorithm for fill the blanks on the grid:

I simply create a map<Integer, Bejeweled> and put one object from each type of math symbols and jewels. So, it looks like (1->Diamond,2->Square so on...). After deleting the triples and shifting it is time to write the grid to screen. While writing when it encounters with empty it replaces the empty coordinate with a randomly selected jewel or math symbol (using Math.random()*map.size())Doing that simply fill the blanks on the grid I thought.

UML DIAGRAM


Powered by yFiles
Triangle
point Integer
getPoint() Integer
toString() String
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>
Diamond
point Integer
getPoint() Integer
toString() String
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>
Jewels
Score(List<Belewled>) List<Belewled>
Square
point Integer
getPoint() Integer
toString() String
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>
Wildcard
point Integer
getPoint() Integer
toString() String
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>
Players
name String
point Integer
getName() String
getPoint() Integer
toString() String
equals(Object) boolean
hashCode() int
equals_detector(List<Players>, Players) boolean
Rank_Writer(List<Players>, int) void
compareTo(Players) int
Main
main(String[]) void
Commands
command_reader(String) void
Belewled
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>
Score(List<Belewled>) List<Belewled>
getPoint() Integer
Grid_writer(List<Map>, int, int) void
Grid_Shifter(List<Map>, int, int) void
Score_Writer(List<Belewled>) int
leaderboard_writer(List<Players>, File) void
BackSlash
toString() String
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>
Pipe
toString() String
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>
Minus
toString() String
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>
MathSymbols
point Integer
getPoint() Integer
Score(List<Belewled>) List<Belewled>
Slash
toString() String
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>
Plus
toString() String
BejeweledDetector(List<Map>, int, int, Integer, Integer, List<Belewled>, Belewled) List<Map>

This is the complete diagram of this Java code.