BBM 104 – INTRODUCTION TO PRGORAMMING LABORATORY II

PROGRAMMING ASSIGNMENT – 3

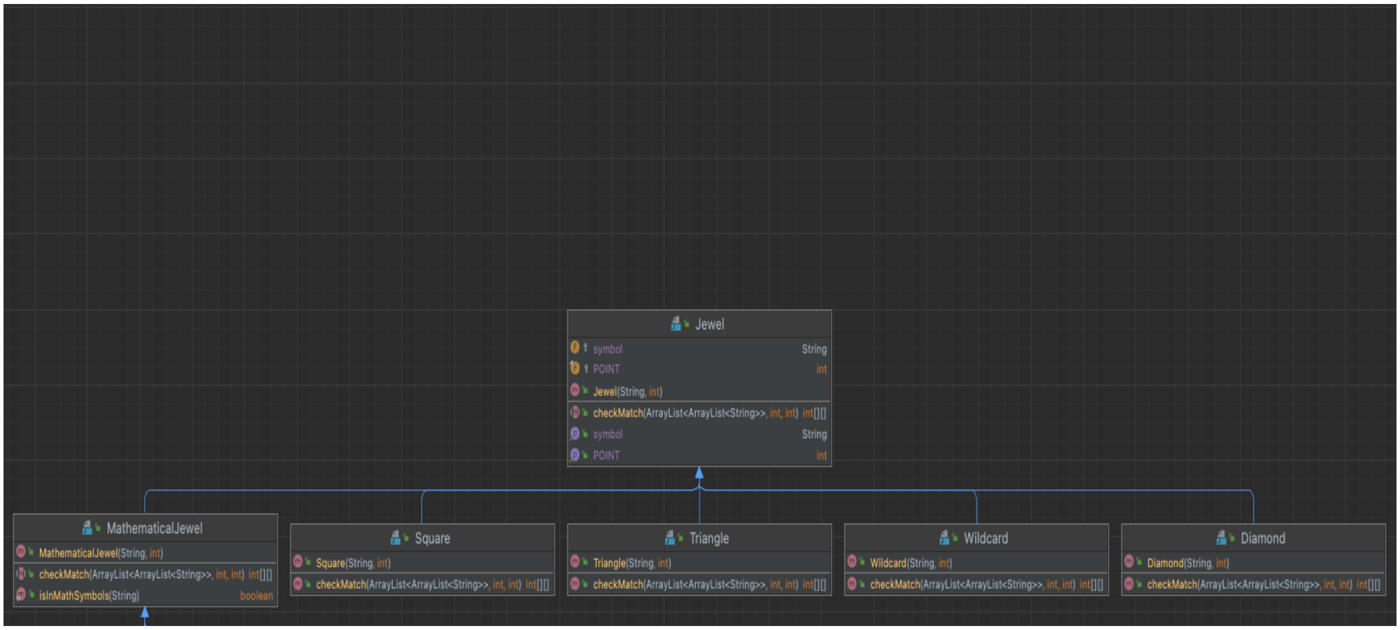
Student Name: Umut GÜNGÖR

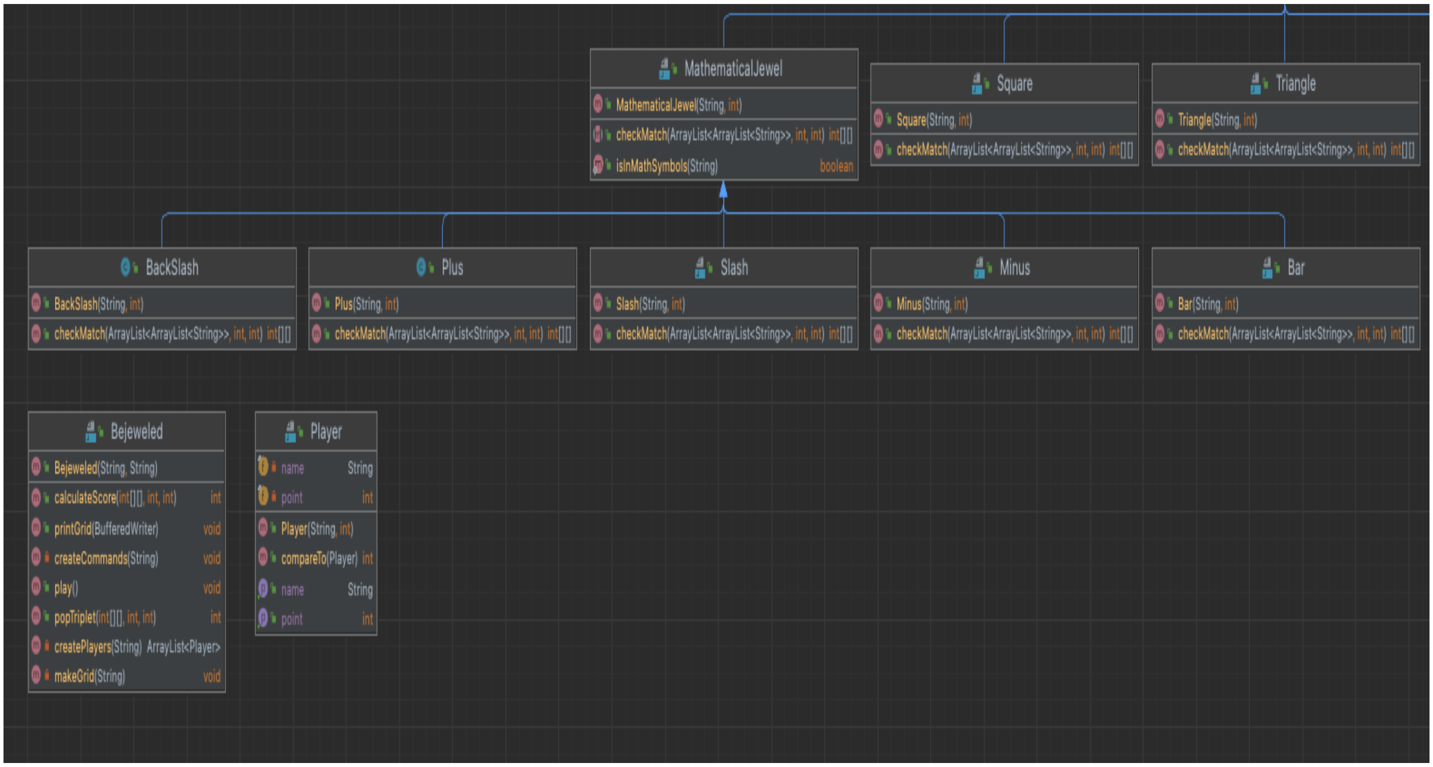
Student Number: 21946198

First of all I created an abstract class called Jewel. It has some fields that all jewels have such as symbol and point. It has getter methods for these fields. Also, it has an abstract method which is named checkMatch. This abstract class is a blueprint for other jewels who inherit from this class. Jewel is a super class and there is five subclasses that are inherited from Jewel class. Square, Triangle, Wildcard and Diamond classes have symbols "S", "T", "W", "D" respectively. These 4 subclasses have their own checkMatch method. Namely, they override that method according to their pop rules. For instance, Diamond class' checkMatch method controls given jewel and its diagonals for finding another Diamond jewels. And the other ones have similar manner. There is one more subclass of Jewel which is MathematicalJewel class. This class is also an abstract class. The purpose of abstractness of this class is that is this class have some subclasses and also it is blueprint for them. It overrides checkMatch method in an abstract way. So, in this class' checkMatch method is also an abstract method. This class has another method which is isInMathSymbols. These method returns a boolean value and checks whether given symbol is mathematical symbol. There are another five classes which are inherited from MathematicalJewel class. They have mathematical symbols. Such that "+", "-", "\" and so on. These classes also override the checkMatch method. They all check mathematical symbols arount them but in their way. For example, Minus class' checkMatch method check the given symbol's right and left sides. The other ones have their own rules for checking the match. With all these classes we are ready to play the game. And there is a game class called Bejeweled. In play method these class starts reading the commands. It is doing the action according to command. It has popTriplet method which is used when there is a match chosen place in the grid. Since every jewel has its own point value we need to calculate the point when there is match by matching jewels. calculateScore method does this job. It calculates the point according to the matching jewels.

When we want to add another jewel to this game what to do is really simple. We create a class for new jewel to differentiate its unique symbol. This class is inherited from Jewel class. It has to override the checkMatch method according to its rules. We add condition cases in Bejeweled class' methods in order to calculate this new jewel's point. After doing these things, new jewel can be used in our game.

UML Diagram


Jewel
symbol String
POINT int
Jewel(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
symbol String
POINT int
MathematicalJewel
MathematicalJewel(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
isInMathSymbols(String) boolean
Square
Square(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
Triangle
Triangle(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
Wildcard
Wildcard(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
Diamond
Diamond(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]


MathematicalJewel
MathematicalJewel(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
isInMathSymbols(String) boolean
Square
Square(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
Triangle
Triangle(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
BackSlash
BackSlash(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
Plus
Plus(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
Slash
Slash(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
Minus
Minus(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
Bar
Bar(String, int)
checkMatch(ArrayList<ArrayList<String>>, int, int) int[][]
Bejeweled
Bejeweled(String, String)
calculateScore(int[][], int, int) int
printGrid(BufferedWriter) void
createCommands(String) void
play() void
popTriplet(int[][], int, int) int
createPlayers(String) ArrayList<Player>
makeGrid(String) void
Player
name String
point int
Player(String, int)
compareTo(Player) int
name String
point int