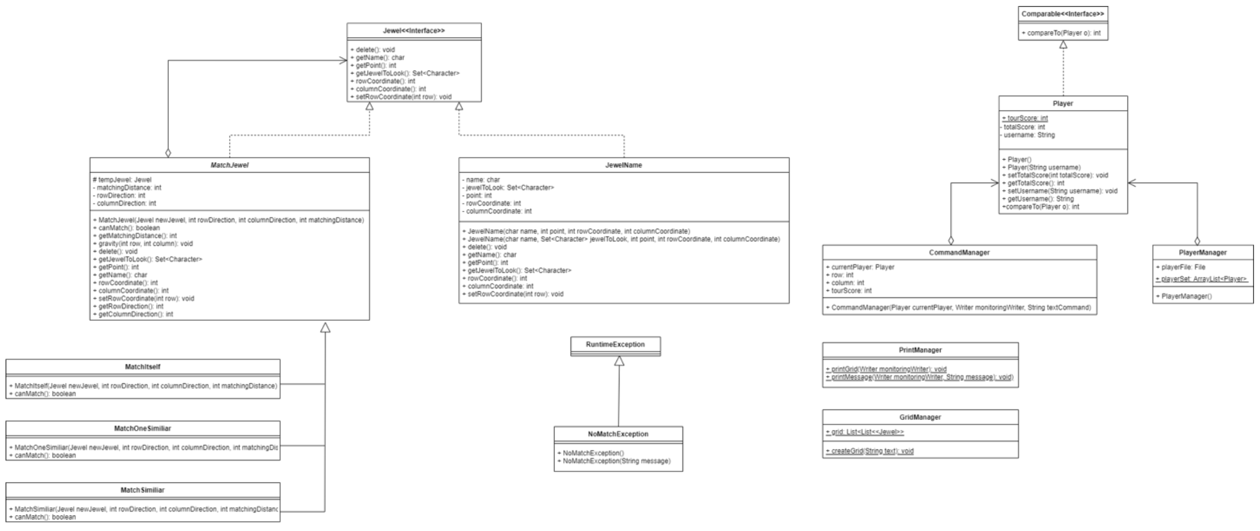
In this assignment, I made a simplified version of the bejeweled game. To do this, I decided to use a decorator pattern because I want to learn this pattern's strong and weak sides. Firstly, I have created a Jewel interface this interface contains methods that we need to use both decorator (MatchJewel) and JewelName class. Secondly, I created a JewelName class for holding Jewel's information like name, point, etc. Every jewel has the same fields, that's why I only created one JewelName class and send everything with the constructor. JewelName's delete method is throwing an exception that shows there is no match because this delete method will run last. Thirdly I started to create decorators. These decorators are MatchItself for only match with itself like 'D'... MatchSimiliar for matching with every same type of symbol like math symbols, and MatchOneSimiliar for matching only one same type symbols like W. If I needed more in the future, I can add new matching possibilities here. Every match decorator has its own canMatch algorithm this algorithm looks if there is a match in a given direction and given length it returns true and deletes method will run. Otherwise, it returns false and tempJewel's match algorithm will run. Fourthly, I created a GridManager class for reading a grid from a text file and creating a static grid list for holding grid objects like a 2D array. The main reason I have used a list is I don't need to specify its length before reading a grid text file. Then I created a player class for holding players' information like username etc. this player implements the comparable interface because of sorting users by its point. I also have PlayerManager class for reading leaderboard text and creating players from this text. CommandManager class is reading a command file and doing operations with them. To conclude, this decorator pattern wasn't fit that much with this assignment because of some reasons. But This homework made me learn every good and bad side of this pattern.


Jewel<<Interface>>
+ delete(): void
+ getName(): char
+ getPoint(): int
+ getJewelToLook(): Set<Character>
+ rowCoordinate(): int
+ columnCoordinate(): int
+ setRowCoordinate(int row): void
MatchJewel
# tempJewel: Jewel
- matchingDistance: int
- rowDirection: int
- columnDirection: int
+ MatchJewel(Jewel newJewel, int rowDirection, int columnDirection, int matchingDistance)
+ canMatch(): boolean
+ getMatchingDistance(): int
+ gravity(int row, int column): void
+ delete(): void
+ getJewelToLook(): Set<Character>
+ getPoint(): int
+ getName(): char
+ rowCoordinate(): int
+ columnCoordinate(): int
+ setRowCoordinate(int row): void
+ getRowDirection(): int
+ getColumnDirection(): int
JewelName
- name: char
- jewelToLook: Set<Character>
- point: int
- rowCoordinate: int
- columnCoordinate: int
+ JewelName(char name, int point, int rowCoordinate, int columnCoordinate)
+ JewelName(char name, Set<Character> jewelToLook, int point, int rowCoordinate, int columnCoordinate)
+ delete(): void
+ getName(): char
+ getPoint(): int
+ getJewelToLook(): Set<Character>
+ rowCoordinate(): int
+ columnCoordinate: int
+ setRowCoordinate(int row): void
MatchItself
+ MatchItself(Jewel newJewel, int rowDirection, int columnDirection, int matchingDistance)
+ canMatch(): boolean
MatchOneSimiliar
+ MatchOneSimiliar(Jewel newJewel, int rowDirection, int columnDirection, int matchingDis
+ canMatch(): boolean
Match Similiar
+ MatchSimiliar(Jewel newJewel, int rowDirection, int columnDirection, int matchingDistanc
+ canMatch(): boolean
RuntimeException
NoMatchException
+ NoMatchException()
+ NoMatchException(String message)
Comparable<<Interface>>
+ compareTo(Player o): int
Player
+ tourScore: int
- totalScore: int
- username: String
+ Player()
+ Player(String username)
+ setTotalScore(int totalScore): void
+ getTotalScore(): int
+ setUsername(String username): void
+ getUsername(): String
+compareTo(Player o): int
CommandManager
+ currentPlayer: Player
+ row: int
+ column: int
+ tourScore: int
+ CommandManager(Player currentPlayer, Writer monitoringWriter, String textCommand)
PlayerManager
+ playerFile: File
+ playerSet: ArrayList<Player>
+ PlayerManager()
PrintManager
+ printGrid(Writer monitoringWriter): void
+ printMessage(Writer monitoringWriter, String message): void)
GridManager
+ grid: List<List<<Jewel>>
+ createGrid(String text): void