Comp 122 Assignment 1 – Matthew Jon Harding 201220275

Part 1 – Requirements

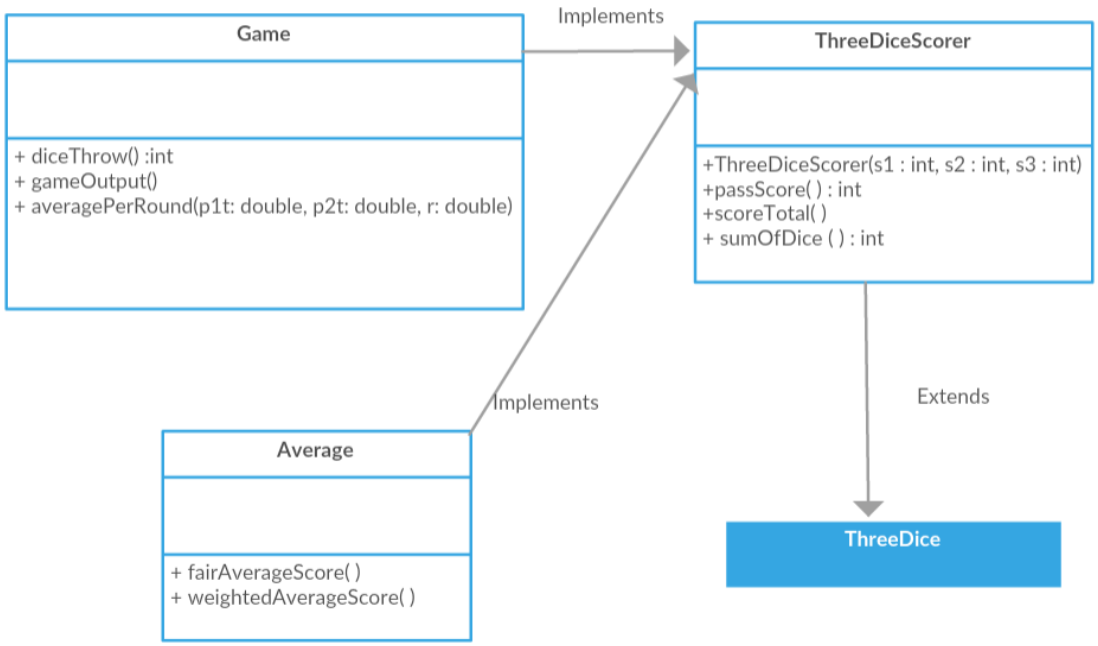
For the first task of this assignment, we are required to design subclasses that extend the provided ThreeDice.java class. We need to build a separate application program that is required to take a user input, in the form of a positive integer or zero, to provide the number of rounds the user wishes to play. The program will need to randomly generate dice rolls resulting in integers from 1 to 6. The game is supposed to have two players, each rolling three dice per round, with a total score given per round at the end with a declaration of which player has won or a draw. At the end of the given number of rounds the program should declare the overall winner by point total, the average number of points obtained by each player, total points, and total round wins.

There is also the additional requirement to create and Average class file which can calculate the average score of for every combination of three fair six-sided die being thrown, and the average score of every combination of 2 fair die and 1 biased die to be thrown with faces [2,3,4,5,6,6].

Part 1 – analysys and design

The class ThreeDiceScorer, will need to have methods to create a ThreeDice object, get the sum of the dice thrown, print out the score and return the value of the score. The Game file should run the main program, create an array for each player, and store the different ThreeDiceScorer objects in for each player on each round.This way if the program needs to be modified in future to check past rounds at some point they arent discarded. Game will also need to take the number of rounds as input from a user and keep a count of the wins, overall score, and average per round and print all this information out at the end.

UML diagrams of ThreeDice and its dependants.



Pseudocode for Game.java

```
For i=0 to i = rounds – 1
  player1[i] = new ThreeDiceScorer
  print “round number player 1 die1 die2 die3” to screen
  print player1[i] score to screen
  player1total += score for this round
  player2[i] = new ThreeDiceScorer
  print “round number Player2 die1 die2 die3” to screen
  print player2[i] score to screen
  player2total += score for this round
  round number + 1

  if player1 score < player 2 print “player 2 wins”
          player2wins += 1
  else if  player1 score == player 2 score
          print “Draw” to screen
  else print “player 1 wins”;
          player1wins += 1
i++
```

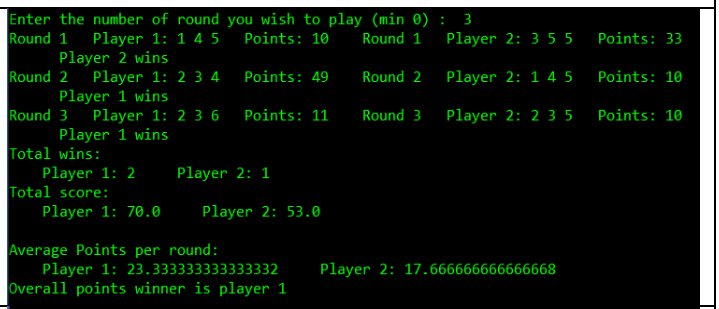
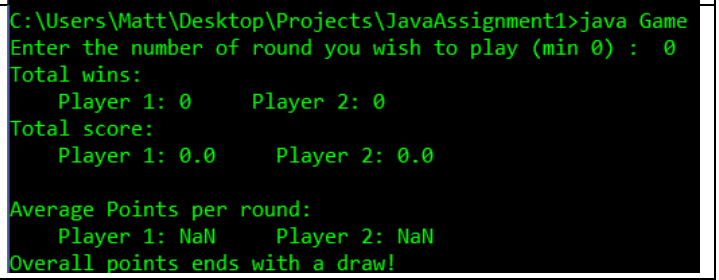
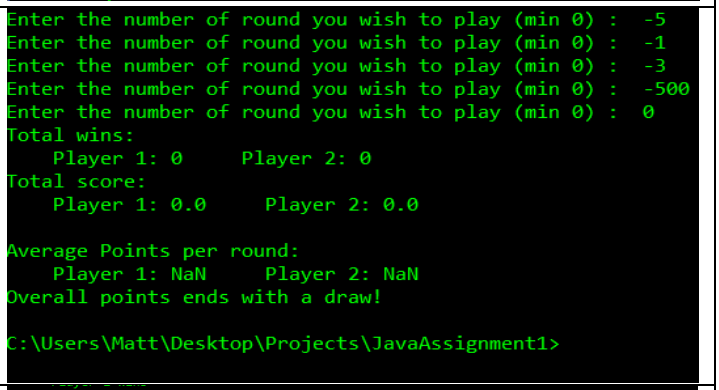
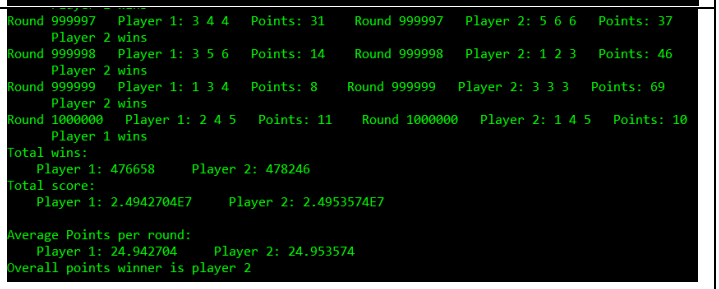
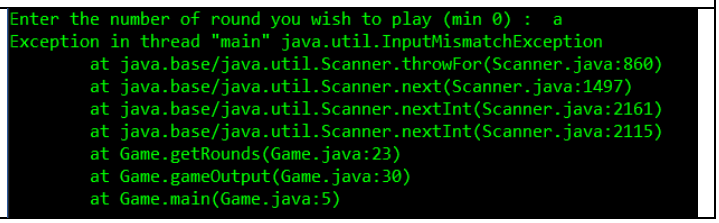
Pseudocode for fair dice score calculation

```
For I = 1 to 6
          For j = 1 to 6
                    For k = 1 to 6
                              Create ThreeDiceScorer with values I J K
                              totalscore = totalscore + score of this threedice object
                    k++
          j++
i++
print total / 216    /*the number of combinations of dice thrown*/
```

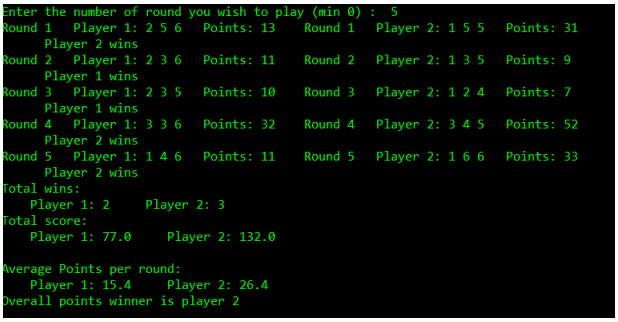
Pseudocode for unfair dice score calculation with faces {2,3,4,5,6,6}

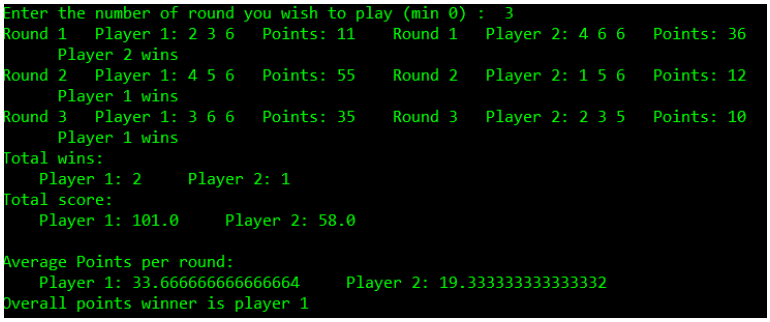
```
For I = 1 to 6
        For j = 1 to 6
                For k = 1 to 6
                        If i == 1 then set i to 6.
                        Create ThreeDiceScorer with values I J K
                        totalscore = totalscore + score of this threedice object
                k++
        j++
i++
print total / 216    /*the number of combinations of dice thrown*/
```

TESTING

INPUT	EXPECTED OUTPUT	Given output
Positive integer of rounds entered (3)	Game played for correct number of rounds, with accurate round counts. Wins and scores.	
Input of Zero rounds	0 rounds play. Program runs without crash or error. Print out 0 scores and ends game in a draw.	
Input of negative round number (-5, -1, -3, -500)	Repeatedly ask for a valid input until one is entered	
Input of an abnormally large number of rounds (1000000)	Rounds played out and average score/wins calculated. Total score may error due to massive number potential.	
Input an invalid entry (‘a’ type – char)	Exception thrown and crashed program.	

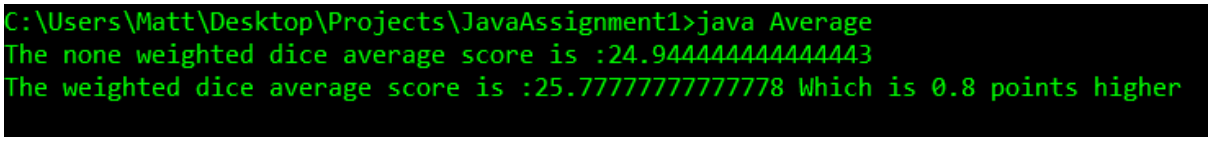
Further game test examples with normal input to check accuracy of code and demonstrate that both players can win a game. A draw is shown in the 0 round test.





Average point calculations

The program I designed to calculate the average amount points for every combination of die thrown for the sets {1,2,3,4,5,6} I refer to as none weighted, and {2,3,4,5,6,6} which I refer to as a weighted set found that there is a 0.8 point increase to the average score, to 1 decimal place.



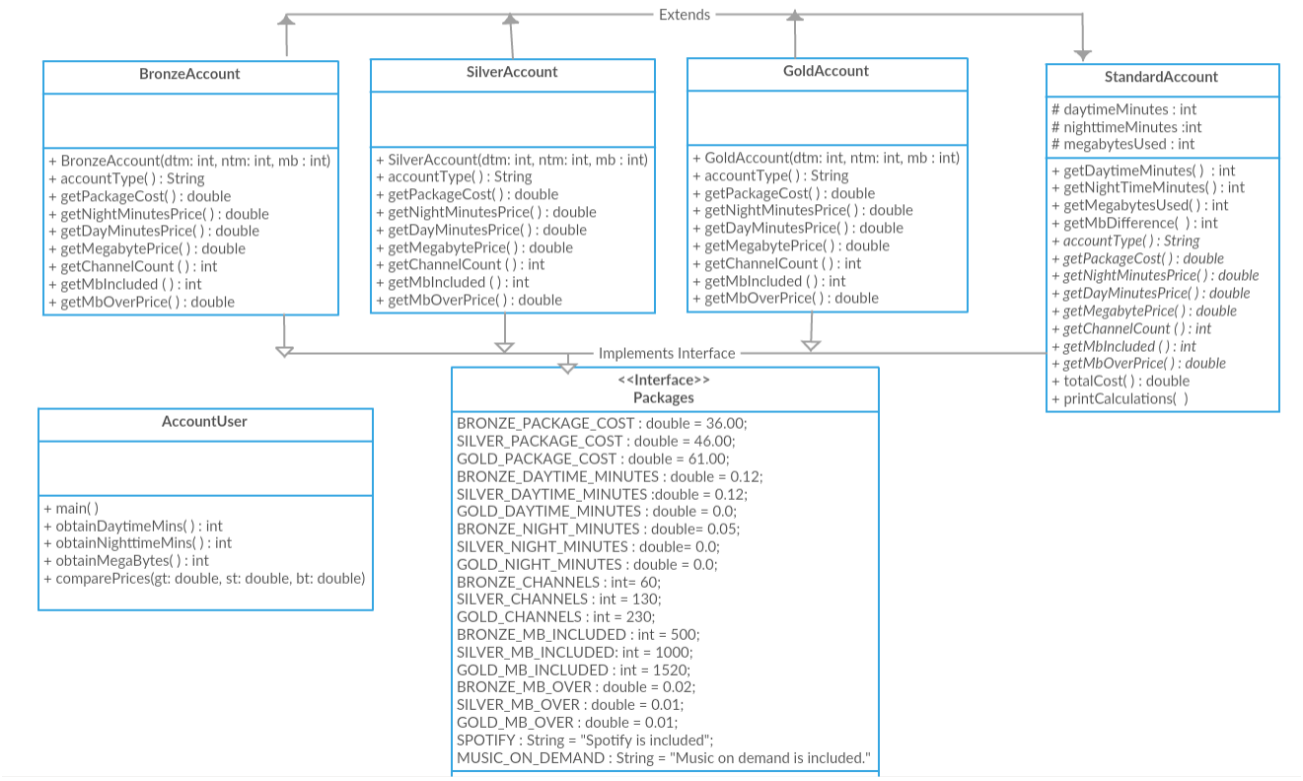
Part 2 – Broadband and minutes

The main requirements for this program were to allow a user to input the amount of daytime and night time minutes into an interface, as well as the amount of broadband data they use per month in Mb. The program should then calculate the cost of certain account packages and determine the cheapest option for the user, or the one with the best value in the case of a tie in cost.

Part 2 analysis and design

The simplest way to implement a solution to this problem is to create relevant account type classes (bronze/ silver/ gold) that extend from an abstract base class, while utilising an interface file to store all the constants. This allows all the calculations to be written once in the “Standard account” abstract class, and for easily changeable constants such as unit price or amounts of a commodity that will impact the rest of the program after being changed once. Each account class will get its relevant constants from the interface file allowing class specific calculations to be done from within the superclass.

UML diagrams



Pseudocode :

AccountUser comparePrices(goldtotal, silvertotal, bronzetotal)

If (goldtotal is less than or equal to silver and bronzetotal)

return gold as cheapest

Else if (silvertotal is less than gold total and less than or equal to bronzetotal)

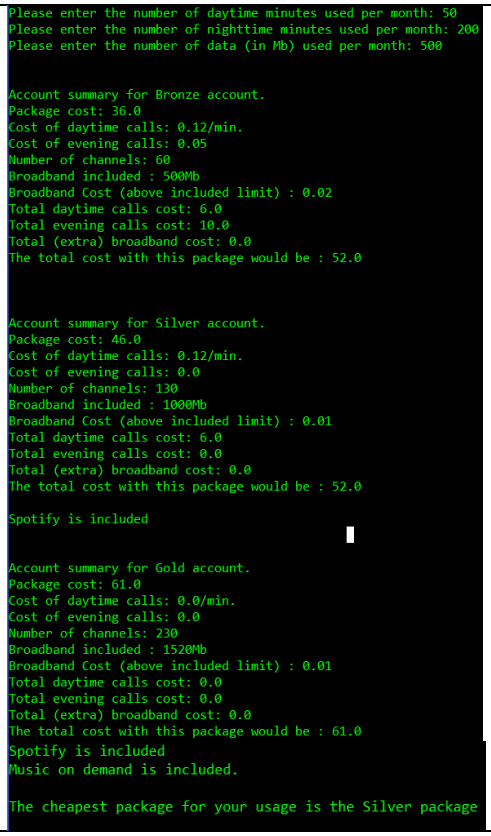
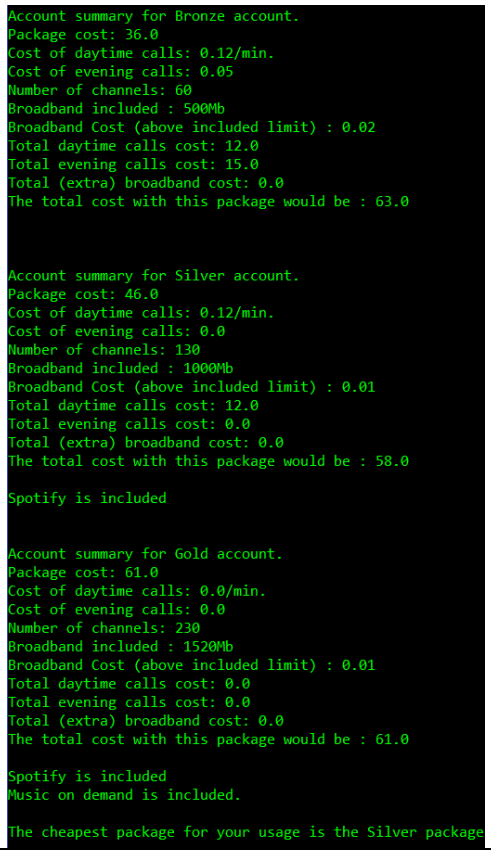
return silver as cheapest.

Else return bronze as cheapest

***(there is no interesting thing else going on other than very simple calculations and multiple print statements)

Testing

INPUT	EXPECTED OUTCOME	RESULT
0 minutes/data	Bronze package is cheapest option / program runs	Account summary for Bronze account. Package cost: 36.0 Cost of daytime calls: 0.12/min. Cost of evening calls: 0.05 Number of channels: 60 Broadband included : 500Mb Broadband Cost (above included limit) : 0.02 Total daytime calls cost: 0.0 Total evening calls cost: 0.0 Total (extra) broadband cost: 0.0 The total cost with this package would be : 36.0 Account summary for Silver account. Package cost: 46.0 Cost of daytime calls: 0.12/min. Cost of evening calls: 0.0 Number of channels: 130 Broadband included : 1000Mb Broadband Cost (above included limit) : 0.01 Total daytime calls cost: 0.0 Total evening calls cost: 0.0 Total (extra) broadband cost: 0.0 The total cost with this package would be : 46.0 Spotify is included Account summary for Gold account. Package cost: 61.0 Cost of daytime calls: 0.0/min. Cost of evening calls: 0.0 Number of channels: 230 Broadband included : 1520Mb Broadband Cost (above included limit) : 0.01 Total daytime calls cost: 0.0 Total evening calls cost: 0.0 Total (extra) broadband cost: 0.0 The total cost with this package would be : 61.0 Spotify is included Music on demand is included. The cheapest package for your usage is the Bronze package
Wrong type of entry (character ‘a’) in each field	Program crash. Exception thrown	Please enter the number of daytime minutes used per month: a Exception in thread "main" java.util.InputMismatchException at java.base/java.util.Scanner.throwFor(Scanner.java:860) at java.base/java.util.Scanner.next(Scanner.java:1497) at java.base/java.util.Scanner.nextInt(Scanner.java:2161) at java.base/java.util.Scanner.nextInt(Scanner.java:2115) at AccountUser.obtainDaytimeMins(AccountUser.java:30) at AccountUser.main(AccountUser.java:6) C:\Users\Matt\Desktop\Projects\JavaAssignment1> Please enter the number of daytime minutes used per month: 1 Please enter the number of nighttime minutes used per month: a Exception in thread "main" java.util.InputMismatchException at java.base/java.util.Scanner.throwFor(Scanner.java:860) at java.base/java.util.Scanner.next(Scanner.java:1497) at java.base/java.util.Scanner.nextInt(Scanner.java:2161) at java.base/java.util.Scanner.nextInt(Scanner.java:2115) at AccountUser.obtainNighttimeMins(AccountUser.java:42) at AccountUser.main(AccountUser.java:7) Please enter the number of daytime minutes used per month: 1 Please enter the number of nighttime minutes used per month: 1 Please enter the number of data (in Mb) used per month: a Exception in thread "main" java.util.InputMismatchException at java.base/java.util.Scanner.throwFor(Scanner.java:860) at java.base/java.util.Scanner.next(Scanner.java:1497) at java.base/java.util.Scanner.nextInt(Scanner.java:2161) at java.base/java.util.Scanner.nextInt(Scanner.java:2115) at AccountUser.obtainMegaBytes(AccountUser.java:55) at AccountUser.main(AccountUser.java:8)

50 day minutes, 200 night minutes, 500 Mb of data. Causing tie in cost between silver and bronze	Recommend silver over bronze as Spotify included.	
100 day minutes, 300 night minutes, 500 mb data entered	Silver clear reccomendation	

125 day minutes, 200 night minutes, 500Mb data (cause tie between silver and gold)	Display gold as cheapest as includes music on demand and Spotify therefore better value package	Account summary for Silver account. Package cost: 46.0 Cost of daytime calls: 0.12/min. Cost of evening calls: 0.0 Number of channels: 130 Broadband included : 1000Mb Broadband Cost (above included limit) : 0.01 Total daytime calls cost: 15.0 Total evening calls cost: 0.0 Total (extra) broadband cost: 0.0 The total cost with this package would be : 61.0 Spotify is included Account summary for Gold account. Package cost: 61.0 Cost of daytime calls: 0.0/min. Cost of evening calls: 0.0 Number of channels: 230 Broadband included : 1520Mb Broadband Cost (above included limit) : 0.01 Total daytime calls cost: 0.0 Total evening calls cost: 0.0 Total (extra) broadband cost: 0.0 The total cost with this package would be : 61.0 Spotify is included Music on demand is included. The cheapest package for your usage is the Gold package
250 day minutes, 250 night minutes, 1500 Mb data.	Gold package calculated as clear cheapest option.	Account summary for Bronze account. Package cost: 36.0 Cost of daytime calls: 0.12/min. Cost of evening calls: 0.05 Number of channels: 60 Broadband included : 500Mb Broadband Cost (above included limit) : 0.02 Total daytime calls cost: 30.0 Total evening calls cost: 12.5 Total (extra) broadband cost: 20.0 The total cost with this package would be : 98.5 Account summary for Silver account. Package cost: 46.0 Cost of daytime calls: 0.12/min. Cost of evening calls: 0.0 Number of channels: 130 Broadband included : 1000Mb Broadband Cost (above included limit) : 0.01 Total daytime calls cost: 30.0 Total evening calls cost: 0.0 Total (extra) broadband cost: 5.0 The total cost with this package would be : 81.0 Spotify is included Account summary for Gold account. Package cost: 61.0 Cost of daytime calls: 0.0/min. Cost of evening calls: 0.0 Number of channels: 230 Broadband included : 1520Mb Broadband Cost (above included limit) : 0.01 Total daytime calls cost: 0.0 Total evening calls cost: 0.0 Total (extra) broadband cost: 0.0 The total cost with this package would be : 61.0 Spotify is included Music on demand is included. The cheapest package for your usage is the Gold package
Negative integer for all 3 entry fields	Re prompt user for input.	Please enter the number of daytime minutes used per month: -60 Value entered must be 0 or above. Please enter the number of daytime minutes used per month: 10 Please enter the number of nighttime minutes used per month: -80 Value entered must be 0 or above. Please enter the number of nighttime minutes used per month: 10 Please enter the number of data (in Mb) used per month: -1000 Value entered must be 0 or above. Please enter the number of data (in Mb) used per month: 60

Implementing a Platinum package (d)

To add a platinum package for this program I would add new Constants to match the style and structure of bronze/silver/gold package but set the PLATINUM_CHANNELS to 275 (int) and MB included, Mb cost, Daytime minute cost and night time minute cost to 0(double). I would then extend the Standard account class to make a PlatinumAccount.java class file and have it mirror the gold/silver/bronze account classes in its structure, making the same overrides to the abstract methods and constructor. I would also set a cost constant as a double at 75.00 .