Java Programming: Practical 1

**Question 1**

* Develop a solution for a Java Class called StockItem.
* The Class contains the following Instance Fields:
  + Item Name (String)
  + Item ID (int)
  + Quantity In Stock (int)
  + Price (double)
* The Class should contain the getters and setters for the above fields.

**Question 2**

* Develop a solution for a Java Class called NumberPair
* The Class contains the following Instance Fields:
  + Number 1 (int)
  + Number 2 (int)
* The Class contains the following Methods:
  + public int sum()
    - Returns the sum of Number 1 + Number 2.
  + public int difference()
    - Returns the result of Number 2 - Number 1.
  + public int product()
    - Returns the result of multiplying Number 1 by Number 2.
  + public double average()
    - Returns the average of Number 1 and Number 2.
  + public int maximum()
    - Returns the larger of the two Values (Number 1 and Number 2).
  + public int minimum()
    - Returns the smaller of the two Values (Number 1 and Number 2).

**Question 3**

* Develop a solution for a Java Class called CurrencyConverter .
* The Class contains the following Instance Fields:
  + Euro to GBP Rate (double)
  + Euro to USD Rate (double)
  + GBP to Euro Rate (double)
  + GBP to USD Rate (double)
  + USD to Euro Rate (double)
  + USD to GBP Rate (double)
* The Class contains the following Methods:
  + public double convertEURtoGBP(int amountToConvert)
    - Converts the specified Euro amount to GBP.
    - Returns the resulting Value.
  + public double convertEURtoUSD(int amountToConvert)
    - Converts the specified Euro amount to USD.
    - Returns the resulting Value.
  + public double convertGBPtoEUR(int amountToConvert)
    - Converts the specified GBP amount to Euro.
    - Returns the resulting Value.
  + public double convertGBPtoUSD(int amountToConvert)
    - Converts the specified GBP amount to USD.
    - Returns the resulting Value.
  + public double convertUSDtoEUR(int amountToConvert)
    - Converts the specified USD amount to Euro.
    - Returns the resulting Value.
  + public double convertUSDtoGBP(int amountToConvert)
    - Converts the specified USD amount to GBP.
    - Returns the resulting Value.

**Question 4**

* Develop a solution for a Java Class called BMICalculator
* The Class contains the following Instance Fields:
  + Weight In KG (double)
  + Height In Metres (double)
* The Class contains the following Methods:
  + public double getBMI()
    - Calculates and returns the Body Mass Index (BMI) .
    - BMI: Weight / (Height \* Height)
  + public String getBMIDescription()
    - Returns a String description of associated BMI.
    - If a BMI Value less than or equal to 18 is specified, the String “Under Weight” should be returned.
    - If a BMI Value less than or equal to 25 is specified, the String “Normal Weight” should be returned.
    - If a BMI Value less than or equal to 30 is specified, the String “Overweight” should be returned.
    - If a BMI Value greater than 30 is specified, “Obese” should be returned.
* When creating the getBMIDescription() Method, you can re-use the code written in the getBMI() Method as follows:

public String getBMIDescription()

{

int bmi = getBMI();

if(bmi <= 18)

{

return “Under Weight”;

}

else if (bmi <= 25)

{

return “Normal Weight”;

}

//etc

}

**Question 5**

* Develop a solution for a Java Class called DataSet
* The Class contains the following Instance Fields:
  + Sum of All Values (int)
  + Number of Values (int)
  + Smallest Value (int)
  + Largest Value (int)
* The Class contains the following Methods:
  + public void addValue(int valueToAdd)
    - Adds the specified Value to the Sum of All Values.
    - Increases the Number of Values by 1.
    - HINT: You should also keep track of both the Smallest and Largest Values within this method.
    - Does not return a Value.
  + public int getSum()
    - Returns the Sum of All Values.
  + public double getAverage()
    - Returns the average for the Sum of All Values.
      * AVERAGE: SUM OF ALL VALUES / Number of Values.
  + public int getLargest()
    - Returns the Largest of all Values entered.
  + public int getSmallest()
    - Returns the Smallest of all Values entered.

**Question 6**

* Develop a solution for a Java Class called FootballGame
* The Class contains the following Instance Fields:
  + Home Team (String)
  + Away Team (String)
  + Home Score (int)
  + Away Score (int)
* The Class contains the following Methods:
  + public void goal(char homeOrAway)
    - Adds one to the score of either the home or the away team depending on whether ‘H’ or ‘A’ is passed into the method.
    - If neither ‘H’ or ‘A’ is passed into the Method, the Method should do nothing.
    - Does not return a Value.
  + public String getScore()
    - Returns the current score in the following String format:  *Manchester United (H) 2 – 1 Liverpool (A)*.

**Question 7**

* Develop a solution for a Java Class called GAA\_Game
* The Class contains the following Instance Fields:
  + Home Team (String)
  + Away Team (String)
  + Home Goals (int)
  + Home Points (int)
  + Away Goals (int)
  + Away Points (int)
* The Class contains the following Methods:
  + public void goal(char homeOrAway)
    - Adds one goal to the score of either the home or the away team depending on whether ‘H’ or ‘A’ is passed into the method.
    - If neither ‘H’ or ‘A’ is passed into the Method, the Method should do nothing.
    - Does not return a Value.
  + public void point(char homeOrAway)
    - Adds one point to the score of either the home or the away team depending on whether ‘H’ or ‘A’ is passed into the method.
    - If neither ‘H’ or ‘A’ is passed into the Method, the Method should do nothing.
    - Does not return a Value.
  + public String getScore()
    - Returns the current score in the following String format: *Dublin (H) 1G7P – 3G14P Donegal (A)*.
  + public String getScoreInPoints()
    - Returns the current score in the following String format (points only): *Dublin (H) 10P – 23P Donegal (A)*.
      * 1 Goal = 3 Points

**Question 8**

* Develop a solution for a Java Class called SnookerGame
* The Class contains the following Instance Fields:
  + Player 1 Name (String)
  + Player 2 Name (String)
  + Player 1 Score (int)
  + Player 2 Score (int)
* The Class contains the following Methods:
  + public void ballPotted(int playerID, char ballColor)
    - ‘R’ => Red = 1 Point
    - ‘Y’ => Yellow = 2 Points
    - ‘G’ => Green = 3 Points
    - ‘B’ => Brown = 4 Points
    - ‘L’ => Blue = 5 Points
    - ‘P’ => Pink = 6 Points
    - ‘A’ => Black = 7 Points
    - Adds the specified points (see above) to the specified players score.
    - If an illegal ball colour is specified, or an illegal player ID is specified, i.e. other than 1 or 2, no action should be taken.
    - Does not return a Value.
  + public String getScore()
    - Returns the current score in the following String format: *Ronnie O’Sullivan 147 – 0 Ken Doherty*.

**Question 9**

* Develop a solution for a Java Class called Car.
* The Class contains the following Instance Fields:
  + Car Make (String), *e.g. Ford*
  + Car Model (String), *e.g. Focus*
  + Car Registration (String)
  + Total Distance Travelled (double)
  + Fuel Tank Size (double)
  + Fuel In Tank (double)
  + Miles Per Litre of Fuel (double)
* The Class contains the following Methods:
  + public String toString()
    - Returns the Car details in the following String format:

Car Make: Ford

Car Model: Focus

Car Registration: 01DL12345

Mileage: 234567

Fuel Tank Size: 40L

Fuel In Tank: 6.7L

Fuel Efficiency: 1 Mile Per Litre

* + public void addFuel(double amountToAdd)
    - Adds the specified amount of fuel to the fuel tank.
    - The amount of fuel specified, along with the amount of fuel already in the tank cannot exceed the Fuel Tank Size.
    - Note: Only amounts greater than zero should be accepted; otherwise, an appropriate error message should be displayed on screen.
    - Does not return a Value.
  + public void drive(double distanceInMiles)
    - Before the Car travels the specified distance, a check should first be made to ensure that there is enough fuel in the tank to travel the specified distance.
    - If there is enough fuel in the tank, the car should travel the specified distance.
      * The amount of fuel in the tank should be decreased appropriately.
      * The mileage of the Car should be increased appropriately.
      * Does not return a Value.

**Question 10**

* Develop a solution for a Java Class called CurrentAccount
* The Class contains the following Instance Fields:
  + Account Owner (String)
  + Account Number (int)
  + Account Balance (double)
  + Overdraft Enabled (boolean)
  + Overdraft Amount (double)
* The Class contains the following Methods:
  + public void lodge(double amountToLodge)
    - Lodges the specified amount to Account.
    - NOTE: An amount to lodge must be greater than zero.
    - Does not return a Value.
  + public double getBalance()
    - Returns the account balance.
  + public void disableOverdraft()
    - Disables the ability to overdraw the account.
    - Does not return a Value.
  + public void enableOverdraft(double overdraftAmount)
    - Enables the account to be overdrawn by the specified amount.
    - Does not return a Value.
  + public void withdraw(double amountToWithdraw)
    - Withdraws the specified amount from the account.
    - NOTE: An amount to withdraw must be greater than zero, and less or equal to the account balance.
      * In the event that an overdraft is enabled on the account, the account can be overdrawn by the amount specified in the overdraftAmount Instance Field.
    - Does not return a Value.