Java Programming: Practical 2

**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)
* Utilise the following code for the Constructor:

public StockItem(String nameIn, int idIn, int qISIn, double priceIn)

{

itemName = nameIn;

itemID = idIn;

quantityInStock = qISIn;

price = priceIn;

}

* The Class contains the following Methods:
  + public int getQuantityInStock()
    - Returns the quantity in stock for the current item.
  + public void addStock(int quantityToAdd)
    - Adds the specified quantity to the quantity stock.
    - Note: Only values greater than 0 should be accepted.
    - Does not return a Value.
  + public void removeStock(int quantityToRemove)
    - Removes the specified quantity from the quantity in stock.
    - Note: Only values greater than 0, but less than or equal to the quantity in stock should be accepted.
    - Does not return a Value.
  + public double getPrice()
    - Returns the price for the current item.
  + public void setPrice(double newPrice)
    - Sets the price of the current item to the price specified.
    - Note: Only values greater than 0 should be accepted.
    - Does not return a Value.
* Once you have created the StockItem Class, create a Class with a main() Method called StockItemProgram.
  + In it, you should ask the user to enter the name of a Stock Item, it’s Stock ID, it’s Quantity in Stock, and it’s Price.
  + Once the user has entered this information, you should then create a StockItem Object.
  + Following this your program should display a menu with the following options:

\*\*\*StockItem Menu\*\*\*

1. Get Quantity In Stock
2. Add Stock
3. Remove Stock
4. Get Price
5. Set Price
6. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 6 which should terminate the Program).

**Question 2**

* Develop a solution for a Java Class called NumberPair
* The Class contains the following Instance Fields:
  + Number 1 (int)
  + Number 2 (int)
* Utilise the following code for the Constructor:

public NumberPair(int no1In, int no2In)

{

number1 = no1In;

number2 = no2In;

}

* 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).
* Once you have created the NumberPair Class, create a Class with a main() Method called NumberPairProgram.
  + In it, you should ask the user to enter two Integer Numbers.
  + Once the user has entered this information, you should then create a NumberPair Object.
  + Following this your program should display a menu with the following options:

\*\*\*NumberPair Menu\*\*\*

1. Sum
2. Difference
3. Product
4. Average
5. Maximum
6. Minimum
7. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 7 which should terminate the Program).

**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)
* Utilise the following code for the Constructor:

public CurrencyConverter(double eToGIn, double eToUIn, double gToEIn, double gToUIn, double uToEIn, double uToGIn)

{

eToG = eToGIn;

eToU = eToUIn;

gToE = gToEIn;

gToU = gToUIn;

uToE = uToEIn;

uToG = uToGIn;

}

* 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.
* Once you have created the CurrencyConverter Class, create a Class with a main() Method called CurrencyConverterProgram.
  + In it, you should ask the user to enter six currency conversion rates.
  + Once the user has entered this information, you should then create a CurrencyConverter Object.
  + Following this your program should display a menu with the following options:

\*\*\*CurrencyConverter Menu\*\*\*

1. Euro To GBP
2. Euro To USD
3. GBP To Euro
4. GBP To USD
5. USD To Euro
6. USD To GBP
7. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 7 which should terminate the Program).

**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)
* Utilise the following code for the Constructor:

public BMICalculator (double weightIn, double heightIn)

{

weight = weightIn;

height = heightIn;

}

* The Class contains the following Methods:
  + public double getBMI()
    - Calculates and returns the Body Mass Index (BMI) for the Values specified in the Constructor.
    - BMI: Weight / (Height \* Height)
  + public String getBMIDescription()
    - Calculates Body Mass Index (BMI) for the Values specified in the Constructor and 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

}

* Once you have created the BMICalculator Class, create a Class with a main() Method called BMICalculatorProgram.
  + In it, you should ask the user to enter their height and weight.
  + Once the user has entered this information, you should then create a BMICalculator Object.
  + Following this your program should display a menu with the following options:

\*\*\*BMICalculator Menu\*\*\*

1. Get BMI
2. Get BMI Description
3. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 3 which should terminate the Program).

**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)
* Utilise the following code for the Constructor:

public DataSet()//Notice That No Values Are Passed In To The Constructor

{

sumOfAllValues = 0;

numberOfValues = 0;

smallestValue = 0;

largestValue = 0;

}

* 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.
* Once you have created the DataSet Class, create a Class with a main() Method called DataSetProgram.
  + In it, you should create a DataSet Object and ensure that each of the aforementioned Methods function correctly.
  + Following this your program should display a menu with the following options:

\*\*\*DataSet Menu\*\*\*

1. Add Value
2. Get Sum
3. Get Average
4. Get Largest
5. Get Smallest
6. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 6 which should terminate the Program).

**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)
* Utilise the following code for the Constructor:

public FootballGame (String homeTeamIn, String awayTeamIn)//Notice That Just Two Values Are Passed In To The Constructor

{

homeTeam = homeTeamIn;

awayTeam = awayTeamIn;

homeScore = 0;

awayScore = 0;

}

* 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)*.
* Once you have created the FootballGame Class, create a Class with a main() Method called FootballGameProgram.
  + In it, you should ask the user to enter the name of Team 1 and Team 2.
  + Once the user has entered this information, you should then create a FootballGame Object.
  + Following this your program should display a menu with the following options:

\*\*\*FootballGame Menu\*\*\*

1. Goal
2. Get Score
3. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 3 which should terminate the Program).

**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)
* Utilise the following code for the Constructor:

public GAA\_Game (String homeTeamIn, String awayTeamIn)//Notice That Just Two Values Are Passed In To The Constructor

{

homeTeam = homeTeamIn;

awayTeam = awayTeamIn;

homeGoals = 0;

awayGoals = 0;

homePoints = 0;

awayPoints = 0;

}

* 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
* Once you have created the GAA\_Game Class, create a Class with a main() Method called GAA\_Game\_Program.
  + In it, you should ask the user to enter the name of Team 1 and Team 2.
  + Once the user has entered this information, you should then create a GAA\_Game Object.
  + Following this your program should display a menu with the following options:

\*\*\*GAA Game Menu\*\*\*

1. Goal
2. Point
3. Display Score
4. Display Score In Points
5. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 5 which should terminate the Program).

**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)
* Utilise the following code for the Constructor:

public SnookerGame (String player1In, String player2In)//Notice That Just Two Values Are Passed In To The Constructor

{

player1 = player1In;

player2 = player2In;

player1Score = 0;

player2Score = 0;

}

* 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*.
* Once you have created the SnookerGame Class, create a Class with a main() Method called SnookerGameProgram.
  + In it, you should ask the user to enter the name of Player 1 and Player 2.
  + Once the user has entered this information, you should then create a SnookerGame Object.
  + Following this your program should display a menu with the following options:

\*\*\*Snooker Game Menu\*\*\*

1. Ball Potted
2. Display Score
3. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 3 which should terminate the Program).

**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)
* Utilise the following code for the Constructor:

public Car (String makeIn, String modelIn, String regIn, double totalDistanceTravelledIn, double fuelTankSizeIn, double fuelInTankIn, double milesPerLitreOfFuelIn)

{

make = makeIn;

model = modelIn;

reg = regIn;

totalDistanceTravelled = totalDistanceTravelledIn;

fuelTankSize = fuelTankSizeIn;

fuelInTank = fuelInTankIn;

milesPerLitreOfFuel = milesPerLitreOfFuelIn;

}

* 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.
* Once you have created the Car Class, create a Class with a main() Method called CarProgram.
  + In it, you should ask the user to enter the Car Make, Car Model, Car Registration, Total Distance Travelled, Fuel Tank Size, Fuel in Tank, and Miles per Litre of Fuel associated with the Car.
  + Once the user has entered this information, you should then create a Car Object and ensure that each of the aforementioned Methods function correctly.
  + Following this your program should display a menu with the following options:

\*\*\*Car Menu\*\*\*

1. To String
2. Add Fuel
3. Drive
4. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 4 which should terminate the Program).

**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)
* Utilise the following code for the Constructor:

public CurrentAccount (String ownerIn, int accNoIn)

{

accOwner = ownerIn;

accNo = accNoIn;

accBal = 0;

overdraftEnabled = false;

overdraftAmount = 0;

}

* 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.
* Once you have created the CurrentAccount Class, create a Class with a main() Method called CurrentAccountProgram.
  + In it, you should ask the user to enter their name and account number.
  + Once the user has entered this information, you should then create a CurrentAccount Object.
  + Following this your program should display a menu with the following options:

\*\*\*CurrentAccount Menu\*\*\*

1. Get Balance
2. Lodge
3. Withdraw
4. Enable Overdraft
5. Disable Overdraft
6. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 7 which should terminate the Program).

**Question 11**

* Develop a solution for a Java Class called SavingsAccount.
* The Class contains the following Instance Fields:
  + Account Owner Name (String)
  + Account Number (int)
  + Account Balance (double)
  + Interest Rate (double)
* Utilise the following code for the Constructor:

public SavingsAccount (String ownerIn, int accNoIn, double interestRateIn)

{

accOwner = ownerIn;

accNo = accNoIn;

accBal = 0;

interestRate = interestRateIn;

}

* The Class contains the following Methods:
  + public double getBalance()
    - Retrieves the Balance of the savings account.
  + public void withdraw(double amountToWithdraw)
    - Withdraws the specified amount from the Savings Account.
    - Note: A Savings Account cannot be overdrawn.
    - Does not return a Value.
  + public void lodge(double amountToLodge)
    - Lodges the specified amount to the Savings Account.
    - NOTE: An amount to lodge must be greater than zero.
    - Does not return a Value.
  + public void applyInterest()
    - Applies the interest rate to the balance of the account for one year.
    - Balance = Balance + (Balance \* (Interest Rate/100)).
    - Does not return a Value.
* Once you have created the SavingsAccount Class, create a Class with a main() Method called SavingsAccountProgram.
  + In it, you should ask the user to enter their name, account number and interest rate.
  + Once the user has entered this information, you should then create a SavingsAccount Object.
  + Following this your program should display a menu with the following options:

\*\*\*SavingsAccount Menu\*\*\*

1. Get Balance
2. Lodge
3. Withdraw
4. Apply Interest
5. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 5 which should terminate the Program).

**Question 12**

* Develop a solution for a Java Class called E\_Mail
* The Class contains the following Instance Fields:
  + Sender (String)
  + Recipient (String)
  + Date Sent (String)
  + Subject (String)
  + Message (String)
* Utilise the following code for the Constructor:

public E\_Mail (String senderIn, String recipientIn, String dateIn, String subjectIn, String messageIn)

{

sender = senderIn;

recipient = recipientIn;

date = dateIn;

subject = subjectIn;

message = messageIn;

}

* The Class contains the following Methods:
  + public String send()
    - Returns a String denoting the contents of the E-Mail in the following format:

**FROM:** [**Shaun.McBrearty@nln.ie**](mailto:Shaun.McBrearty@nln.ie)

**TO:** [**Paul.Herron@nln.ie**](mailto:Paul.Herron@nln.ie)

**Date: 04/10/2015**

**SUBJECT: Software Developer Course**

**MESSAGE: 4 weeks in, 34 to go!!!**

* + public String rescind()
    - Returns a String outlining that the previous E-Mail has been rescinded. The String is returned in the following format:

**The Sender,** [**Shaun.McBrearty@nln.ie**](mailto:Shaun.McBrearty@nln.ie)**, would like rescind the previous message.**

**MESSAGE RESCINDED:**

**FROM:** [**Shaun.McBrearty@nln.ie**](mailto:Shaun.McBrearty@nln.ie)

**TO:** [**Paul.Herron@nln.ie**](mailto:Paul.Herron@nln.ie)

**Date: 04/10/2015**

**SUBJECT: Software Developer Course**

**MESSAGE: 4 weeks in, 34 to go!!!**

* When creating the rescind() Method, you can re-use the code written in the send() Method as follows:

public String rescind()

{

String rescindString = “…MESSAGE RESCINDED:/n/n” + send();

//etc

}

* Once you have created the E\_Mail Class, create a Class with a main() Method called E\_MailProgram.
  + In it, you should ask the user to enter their e-mail address, the recipients e-mail address, the date, the e-mail subject, and the e-mail message they wish to send.
  + Once the user has entered this information, you should then create an E\_Mail Object.
  + Following this, you should execute the send() method on the E\_Mail Object created previously and display the resulting String to the Console.
  + Following this, you should then ask the user if they wish to rescind the message.
    - If the user enters ‘Yes’, the rescind() method should be executed (with the resulting String being displayed to the Console).

**Question 13**

* Develop a solution for a Java Class called Employee
* The Class contains the following Instance Fields:
  + Employee Number (int)
  + Employee Name (String)
  + PPS Number (String)
  + Job Title (String)
  + Salary (double)
* Utilise the following code for the Constructor:

public Employee (int employeeNoIn, String nameIn, String ppsIn, String jobIn, double salaryIn)

{

employeeNo = employeeNoIn;

name = nameIn;

pps = ppsIn;

job = jobIn;

salary = salaryIn;

}

* The Class contains the following Methods:
  + public double getSalary()
    - Returns the Employee’s salary.
  + public void payRise()
    - If the Employee’s current salary is less than €50,000, €5000 should be added on to their existing salary; otherwise €2,500 should be added on to their existing salary.
    - Does not return a Value.

* Once you have created the Employee Class, create a Class with a main() Method called EmployeeProgram.
  + In it, you should ask the user to enter their employee number, employee name, pps number, job title and salary.
  + Once the user has entered this information, you should then create a Employee Object.
  + Following this your program should display a menu with the following options:

\*\*\*Employee Menu\*\*\*

1. Get Salary
2. Pay Rise
3. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 3 which should terminate the Program).

**Question 14**

* Develop a solution for a Java Class called Address
* The Class contains the following Instance Fields:
  + Address Line 1 (String)
  + Address Line 2 (String)
  + Address Line 3 (String)
  + Town (String)
* Utilise the following code for the Constructor:

public Address (String add1In, String add2In, String add3In, String townIn)

{

add1 = add1In;

add2 = add2In;

add3 = add3In;

town = townIn;

}

* The Class contains the following Methods:
  + public String toString()
    - Return the Address Details in String format as follows:

Add Line #1: 45 Ballymacool Park

Add Line #2: Long Lane

Add Line #3:

Town: Letterkenny

* Once you have created the Address Class, create a Class with a main() Method called AddressProgram.
  + In it, you should ask the user to enter their address line 1, address line 2, address line 3, and town.
  + Once the user has entered this information, you should then create an Address Object.
  + Following this your program should display a menu with the following options:

\*\*\*Address Menu\*\*\*

1. To String
2. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.

* After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 2 which should terminate the Program).

**Question 15**

* Go to the shared drive and retrieve the folder titled ‘Lesson 15’.
* Within this folder, there is another folder called ‘Q15’.
* ‘Q15’ is a Java Project – Import it into Eclipse.
* Examine the code coatained within ‘Q15’. Lookout for the following within the source code:
  + The Main Class contains two Employee Objects.
  + The Employee Class contains an Address Object as one of its Attributes.
  + The Employee Class has two different Constructors.
    - One That Accepts An Address Object In The Constructor
    - One That Creates An Address Object Using Values Passed Into The Employee Constructor.
  + Calling the toString() Method on the Employee Object results in the toString() Method in the Address Object being called also.

**Question 16**

* Develop a solution for a Java Class called Television
* The Class contains the following Instance Fields:
  + Minimum Channel (int)
  + Maximum Channel (int)
  + Current Channel (int)
* Utilise the following code for the Constructor:

public Television()

{

minimumChannel = 1;

currentChannel = 1;

maximumChannel = 100;

}

* The Class contains the following Methods:
  + public void plusOne()
    - Increases current channel by one.
    - Should current channel equal maximum channel, the value of current channel should wrap around to one again.
    - Does not return a Value.
  + public void minusOne()
    - Decreases current channel by one.
    - Should current channel equal minimum channel, the value of current channel should wrap around to onehundred.
    - Does not return a Value.
  + public void setChannel(int channelNo)
    - Sets the value of current channel to channelNo.
    - Note: Only values greater than or equal to minimum channel, but less than or equal to maximum channel should be accepted.
    - Does not return a Value.
  + public int getCurrentChannel()
    - Returns the current channel number in integer form.

* Once you have created the Television Class, create a Class with a main() Method called Television Program.
  + The program should display a menu with the following options:

\*\*\* Television Menu\*\*\*

1. Get Current Channel
2. Plus One
3. Minus One
4. Set Channel
5. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 5 which should terminate the Program).

**Question 17**

* Develop a solution for a Java Class called WaterContainer
* The Class contains the following Instance Fields:
  + Maximum Capacity (double)
  + Current Capacity (double)
  + Drainage Speed Per Second (double)
* Utilise the following code for the Constructor:

public WaterContainer (double maximumCapacityIn, double currentCapacityIn, double drainageSpeedPerSecondIn)

{

maximumCapacity = maximumCapacityIn;

currentCapacity = currentCapacityIn;

drainageSpeedPerSecond = drainageSpeedPerSecondIn;

}

* The Class contains the following Methods:
  + public double getCurrentCapacity()
    - Returns the amount of water currently in the WaterContainer.
  + public double getMaximumCapacity()
    - Returns the maximum capacity of the WaterContainer.
  + public void addWater(double quantityToAdd)
    - Adds the specified amount of Water to the Current Capacity of the WaterContainer.
    - If the sum of Current Capacity and quantityToAdd exceeds the Maximum Capacity of the Water Container, the quantityToAdd should simply be ignored.
    - Does not return a Value.
  + public double removeWater(double quantityToRemove)
    - Removes the specified amount of Water from the Current Capacity of the WaterContainer.
    - If the result of subtracting quantityToRemove from Current Capacity is less than zero, the quantityToRemove should simply be ignored.
    - The method should return the time in seconds to remove the amount of water specified (Time to Remove = Quantity To Remove/Drainage Speed).

* Once you have created the WaterContainer Class, create a Class with a main() Method called WaterContainerProgram.
  + In it, you should ask the user to enter the maximum capacity of the Water Container, the current Capacity of the Water Container and the drainage speed per second of the Water Container, *i.e. how many litres of water can be removed from the container in a second*.
  + Once the user has entered this information, you should then create a WaterContainer Object.
  + Following this your program should display a menu with the following options:

\*\*\* WaterContainer Menu\*\*\*

1. Get Maximum Capacity
2. Get Current Capacity
3. Add Water
4. Remove Water
5. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 5 which should terminate the Program).

**Question 18**

* Develop a solution for a Java Class called CalorieCounter
* The Class contains the following Instance Fields:
  + Maximum Calories Per Day (int)
  + Number Of Calories (int)
  + Number Of Meals (int)
  + Highest Calorie Meal (int)
  + Lowest Calorie Meal (int)
* Utilise the following code for the Constructor:

public CalorieCounter(int maxCaloriesPerDayIn)

{

maxCaloriesPerDay = maxCaloriesPerDayIn;

numberOfCalories = 0;

numberOfMeals = 0;

highestCalorieMeal = 0;

lowestCalorieMeal = 0;

}

* The Class contains the following Methods:
  + public int getMaximumCaloriesPerDay()
    - Returns the maximum calories the user can consume per day.
  + public int getNumberOfMealsConsumed()
    - Returns the number of meals consumed by the user for the given day.
  + public int getHighestCalorieMeal()
    - Returns the number of calories contained within the highest calorific meal consumed by the user for the given day.
  + public int getLowestCalorieMeal()
    - Returns the number of calories contained within the lowest calorific meal consumed by the user for the given day.
  + public void addMeal(int caloriesIn)
    - Adds caloriesIn to number of calories.
      * Values less than zero should be ignored by the method.
    - Increases number of meals by one.
    - Updates highest/lowest calorie meal if necessary.
  + public void reset()
    - Resets the value of all Attributes/Properties to zero.
  + public String toString()
    - Returns a String outlining the user’s progress against the target calorie count.
    - The String returned should be of the following format (Note that the value in brackets to the right of Number of Calories Consumed Denotes How Much Below Or Above The User Is In Relation To Their Maximum Calories Per Day Setting):

Maximum Calories Per Day: 2500

Meals Consumed: 2

Number Of Calories Consumed: 1000 (-1500)

Highest Calorie Meal: 600

Lowest Calorie Meal: 400

* Once you have created CalorieCounter Class, create a Class with a main() Method called CalorieCounterProgram.
  + In it, you should ask the user to enter their calorie goal for the day.
  + Once the user has entered this information, you should then create a CalorieCounter Object.
  + Following this your program should display a menu with the following options:

\*\*\* CalorieCounter Menu\*\*\*

1. Get Maximum Calories Per Day
2. Add Meal
3. Get Number Of Meals Consumed
4. Get Highest Calorie Meal Consumed
5. Get Lowest Calorie Meal Consumed
6. Reset
7. To String
8. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 8 which should terminate the Program).

**Question 19**

* Develop a solution for a Java Class called Treadmill
* The Class contains the following Instance Fields:
  + Minimum Speed (double)
  + Maximum Speed (double)
  + Current Speed (double)
  + Running (boolean)
* Utilise the following code for the Constructor:

public Treadmill()

{

minimumSpeed = 1.5;

maximumSpeed = 10;

currentSpeed = 0;

running = false;

}

* The Class contains the following Methods:
  + public void start()
    - Sets running attribute to true.
    - Sets current speed to minimum speed.
    - Does not return a Value.
  + public void stop()
    - Sets running attribute to false.
    - Sets current speed to zero.
    - Does not return a Value.
  + public void increaseSpeed()
    - Increases current speed by 0.1.
      * Note that the treadmill must be running first in order to increase the speed.
      * Note that the speed of the treadmill cannot exceed the value of maximum speed.
    - Does not return a Value.
  + public void decreaseSpeed()
    - Decreases current speed by 0.1.
      * Note that the treadmill must be running first in order to decrease the speed.
      * Note that the speed of the treadmill cannot be below the value of minimum speed.
    - Does not return a Value.
  + public void setSpeed(double speedIn)
    - Sets current speed to the value of speedIn.
      * Note that the treadmill must be running first.
      * Note that the speed of the treadmill cannot be below the value of minimum speed, or above the maximum speed.
    - Does not return a Value.
  + public double getCurrentSpeed()
    - Returns a double representing the current speed that the treadmill is travelling at.
  + public double getMinimumSpeed()
    - Returns a double representing the minimum speed that the treadmill can travel at.
  + public double getMaximumSpeed()
    - Returns a double representing the maximum speed that the treadmill can travel at.
  + public double getCaloriesBurned(double distanceCoveredInKM, double weightInKG, double heightInCM)
    - Utilising the value assigned to current speed, the method returns a double outlining the number of calories burned by a user (according to the distance they’ve ran, their weight in KG and their height in CM).
    - Calories Burned = (Height In CM2 / Weight In KG2) \* Distance Covered In KM \* Current Speed

* Once you have created Treadmill Class, create a Class with a main() Method called TreadmillProgram.
  + The program should display a menu with the following options:

\*\*\* Treadmill Menu\*\*\*

1. Start
2. Stop
3. Increase Speed
4. Decrease Speed
5. Set Speed
6. Get Current Speed
7. Get Maximum Speed
8. Get Minimum Speed
9. Get Calories Burned
10. Quit
    * Whatever menu option the user chooses, the associated Method should then be executed.
    * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 10 which should terminate the Program).

**Question 20**

* Develop a solution for a Java Class called Date
* The Class contains the following Instance Fields:
  + Day (int)
  + Month (int)
  + Year (int)
* Utilise the following code for the Constructor:

public Date(int dayIn, int monthIn, int yearIn)

{

day = dayIn;

month = monthIn;

year = YearIn;

}

* The Class contains the following Methods:
  + public boolean validDate()
    - Returns a Boolean value, *i.e. true or false*, outlining whether or not the value assigned to the date Object is valid or not.
    - HINT: Ensure that your method takes into account leap years.
  + public int getJulianDate()
    - Returns an integer representing the Date in Julian format.
    - e.g. 01/02/2015 = 32, *i.e. 32nd Day In Year*
  + public String addDate(int noOfDays)
    - * Adds the specified number of days to the current date.
      * In the event that the value specified is less than one, the value should simply be ignored.
    - *e.g. 01/02/2015 + 9 = 10/02/2015*
    - Returns a String representing the resulting date.
      * The String should be in the following format: DD/MM/YYYY.
    - HINT: You may want to get the current date as a Julian Date, add on the specified number of days, and then convert the resulting value back to day, month and year form.
    - Note that care should be taken where the resulting date causes the month to change, *e.g. 31/03/2015 +1 = 1/****04****/2015*, or where the resulting date causes the month and the date to change, *31/12/2015 +1 = 1/****01/2016****.*
  + public String subtractDate(int noOfDays)
    - * Subtracts the specified number of days from the current date.
        + In the event that the value specified is less than one, the value should simply be ignored.
    - *e.g. 10/02/2015 - 9 = 01/02/2015*
    - Returns a String representing the resulting date.
      * The String should be in the following format: DD/MM/YYYY.
    - HINT: You may want to get the current date as a Julian Date, subtract the specified number of days, and then convert the resulting value back to day, month and year form.
  + public String toShortDate()
    - * Returns the current date as a String of the following form: *DD/MM/YY*.
  + public String toLongDate()
    - * Returns the current date as a String of the following form: *DD Month Name YY*.
      * Month Name = *January, February, March, etc*.
* Once you have created Date Class, create a Class with a main() Method called DateProgram.
  + In it, you should ask the user to enter the day, month and year in integer format.
  + Once the user has entered this information, you should then create a Date Object.
  + Following this your program should display a menu with the following options:

\*\*\* Date Menu\*\*\*

1. Valid Date
2. Get Julian Date
3. Add Date
4. Subtract Date
5. To Long Date
6. To Short Date
7. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 7 which should terminate the Program).

**Question 21**

* Develop a solution for a Java Program that will automate climate control within a smart home.
* The Program will consist of the following Java Classes:
  + Radiator
  + AirCon
  + AirVent
  + SmartHome
* The Radiator Class will be structured as follows:
  + Attributes:
    - running (Boolean)
  + Actions:
    - public void turnOn()
      * Sets Running Attribute to True.
      * Does not return a Value.
    - public void turnOff()
      * Sets Running Attribute to False.
      * Does not return a Value.
    - public String toString()
      * Returns a String outlining the Radiators Status.
      * String returned should be one of the following:

*Radiator: Running*

*Radiator: Not Running*

* + Constructor
    - The Constructor for the Radiator Class will assign false to the running attribute.
* The AirCon Class will be structured as follows:
  + Attributes:
    - running (Boolean)
  + Actions:
    - public void turnOn()
      * Sets Running Attribute to True.
      * Does not return a Value.
    - public void turnOff()
      * Sets Running Attribute to False.
      * Does not return a Value.
    - public String toString()
      * Returns a String outlining the AirCon Status.
      * String returned should be one of the following:

*AirCon: Running*

*AirCon: Not Running*

* + Constructor
    - The Constructor for the AirCon Class will assign false to the running attribute.
* The AirVent Class will be structured as follows:
  + Attributes:
    - Status (String)
  + Actions:
    - public void open()
      * Sets Status to “Open”.
    - public void halfOpen()
      * Sets Status to “Half-Open”.
    - public void close()
      * Sets Status to “Closed”.
    - public String toString()
      * Returns a String outlining the AirVent’s Status.
      * String returned should be one of the following:

*AirVent: Open*

*AirVent: Half-Open*

*AirVent: Closed*

* + Constructor
    - The Constructor for the AirVent Class will assign “Closed” to the Status attribute.
* The SmartHome Class will be structured as follows:
  + Attributes:
    - Temperature (double)
    - Radiator 1 (Radiator Object)
    - Radiator 2 (Radiator Object)
    - Air Con (AirCon Object)
    - Air Vent (AirVent Object)
  + Actions:
    - public void setTemperature(double tempIn)
      * If the temperature entered by the user is less than or equal to 10 degrees, both radiators should be turned on, the air conditioning should be turned off, and the air vents should be closed.
      * If the temperature entered by the user is between 10 and 20 degrees, both radiators should be turned off, the air conditioning should be turned on, and the air vents should be set to half open.
      * If the temperature entered by the user exceeds 20 degrees, both radiators should be turned off, the air conditioning should be turned on, and the air vents should be set to open.
    - public String toString()
      * Returns a String outlining the Smart Home’s Status.
      * String returned should be of the following format:

*Temperature: 7*

*#1 Radiator: Running*

*#2 Radiator: Running*

*AirCon: Not Running*

*AirVent: Closed*

* + Constructor
    - The Constructor for the SmartHome Class will create two Radiator Objects, one AirCon Object, one AirVent Object, and assign a value to temperature that is specified via a Constructor Paramater.
* Once you have created the four classes, create a Class with a main() Method called SmartHomeProgram.
  + In it, you should ask the user to the enter the current temperature in the house.
  + Once the user has entered this information, you should then create a SmartHome Object.
  + Following this your program should display a menu with the following options:

\*\*\*SmartHome Menu\*\*\*

1. Set Temperature
2. To String
3. Quit
   * Whatever menu option the user chooses, the associated Method should then be executed.
   * After each Method has finished executing, the above menu should be redisplayed (with the exception of Option 3 which should terminate the Program).