Question 1

Create a class TaxCaculator that will bundle together several static methods for tax computations. This class should not have a constructor. Its attributes are

basicRate - the basic tax rate as a static double variable that starts at 4 percent
luxuryRate - the luxury tax rate as a static double variable that starts at 10 percent

Its methods are

- computeCostBasic(price)—a static method that returns the given price plus the basic tax, rounded to the nearest 2 decimal places.
- computeCostLuxury(price)—a static method that returns the given price plus the luxury tax, rounded to nearest 2 decimal places.
- changeBasicRateTo(newRate)—a static method that changes the basic tax rate.
- changeLuxuryRateTo(newRate)—a static method that changes the luxury tax rate.
- roundToNearestTwoDecPlaces(price)—a private static method that returns the given price rounded to the nearest 2 decimal places. For example, if the price is 12.567, the method will return 12.57.

Qustion 2

The EWallet class diagram:

EWallet
- serialNumber: String
- balance: double
- maxAmount: double
+EWallet(String serialNumber, double balance, double maxAmount)
+EWallet(String serialNumber, double maxAmount)
+ getSerialNumber():String
+ getBalance():double
+ topUp(double amount):void
+ deduct(double amount):void
+ isEmpty():Boolean
- canTopUp(double amount):boolean
+ display():void

Name: EWallet
Attributes:

- Serial Number, String data type, private
- Balance, double data type, private
- Max Amount, double data type, private

Constructors:

- First constructor: initialise all the three attributes
- First constructor: initialise the two attributes and set balance to 0

Methods:

- A getSerialNumber method that returns the Serial Number (public)
- A getBalance method that returns the Balance(public)
- A topUp method that takes in a double amount (public)
- A deduct method that takes in a double amount (public)
- A isEmpty method that returns a boolean (public)
- A canTopUp method that takes in a double amount and return a boolean. (private)
- A display method. (public)

1. Create the EWallet class with the Attributes only.

2. Write the display and getBalance method. Write a main() to test the methods. (We may set the serialNumber and balance to public for now to test the method)

3. Write the isEmpty method. The method returns true is balance is 0, else it returns false. Write a main() to test the method. (We may set the balance to public for now to test the method)

4. Write the getSerialNumber method. This method returns the Serial Number. Write a main() to test the method.

5. Write the canTopUp private method. This method takes in a top up amount, it returns a true if the sum of current and top up amount is less than Max Amount. Else if returns false. Write a main() to test the method.

6. Write the topUp method. This method takes in a top up amount and add it to current balance. It uses the canTopUp method to check if it is able to top up.
   It prints a successful or error message. Write a main() to test the method.

7. Write the deduct method. Write a main() to test the method. Check if there is sufficient amount to be deducted. It prints a successful or error message.

8. Write a new Java class and create two EWallet objects with the following initial information:

   EWallet 1
   - Serial Number: T456
   - Balance: 10
   - Max Amount: 100

   EWallet 2
   - Serial Number: T890
   - Balance: 0
   - Max Amount: 500

   Perform the following transaction
   Top up eWallet 1 with $30
   Deduct eWallet 1 with $10
   Top up eWallet 2 with $3000
   Deduct eWallet 2 with $3000

   Create another two eWallet objects with the same initial information.
   - Serial Number: T456
   - Balance: 10
   - Max Amount: 100

   Use == to compare the two eWallet objects and print the outcome.
   Explain the outcome by drawing the memory map sequence of the above codes.

9. Implements a equals() method for EWallet object.
   The two objects are considered equal if the two objects Serial Number and Balance are the same.
   (Note: how two objects are consider "equals" depends on how we define the "equals")

10. Create 5 EWallet object and store them in an ArrayList.

11. Find and display the EWallet(s) with the highest and lowest balance.

12. Create a few more EWallet object s in the ArrayList and sort the EWallet objects by Serial Number.

13. User enter an EWallet serial number and display the EWallet information.