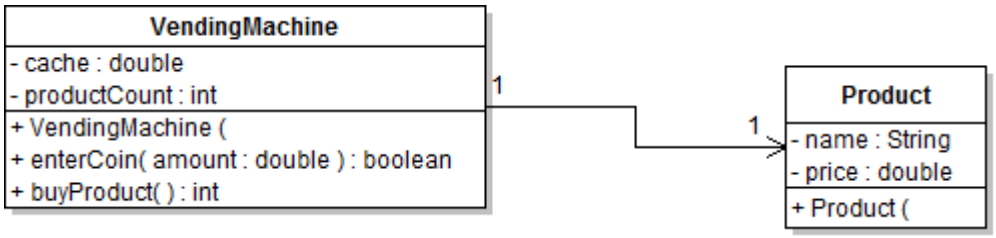
QUESTIONS

Answer the questions according to the incomplete UML class schema below. You will be writing code for a vending machine which sells only one type of product and does not return any change. However, a person can buy more than one product if enough money has been entered. Keep in mind that the class schema is incomplete. Therefore you will need to add necessary members to classes. Take necessary precautions to prevent illogical cases while coding.



Question 1: Write the source code of class Product.

Question 2: Write the source code of class VendingMachine. In order to get a product from the machine, a user must first enter coins into the machine by using the enterCoin method. This method must only accept the following coins: 0.10, 0.25, 0.50 and 1.00. The entered coins are stored in the cache member. After the worth of coins in the cache reaches or exceeds the price of the product, the user can buy the product by using the buyProduct method. The buyProduct returns the given item count.

Question 3: Write a new class with a main method for testing the classes you have written so far. Taking user input is necessary. Moreover, using the enterCoins and buyProduct methods are required.

Question 4: Draw the sequence diagram of the main method you have coded in question 3. Include the calls to obtain user input and do not forget to draw nested method calls.

Question 1: Write the source code of class Product. (10p)

```
public class Product {
    private String name;
    private double price;
    public Product(String name, double price) {
        this.name = name; this.price = price;
    }
    public String getName() { return name; }
    public double getPrice() { return price; }
}
```

Question 2: Write the source code of class VendingMachine. (35p)

```
public class VendingMachine {
    private Product product;
    private double cache;
    private int productCount;
    public VendingMachine(Product product, int productCount) {
        this.product = product; this.productCount = productCount;
        cache = 0.0;
    }
    public boolean enterCoin( double amount ) {
        if( amount == 0.10 || amount == 0.25 ||
                amount == 0.50 || amount == 1.00 ) {
            cache += amount;
            return true;
        }
        return false;
    }
    public int buyProduct() {
        int givenItemCount = (int)(cache/product.getPrice());
        cache = 0.0;
        if( productCount >= givenItemCount) {
            productCount -= givenItemCount;
        }
        else {
            givenItemCount = productCount;
            productCount = 0;
```

```
        }
        return givenItemCount;
    }
    public int getProductCount() { return productCount; }
    // no setters must be coded for cache and productCount!
}
```

Question 3: Write a new class with a main method for testing the classes you have written so far. (35p)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter a product name: ");
        String name = in.nextLine();
        System.out.print("Enter its cost: ");
        double cost = in.nextDouble();
        Product product = new Product(name,cost);
        System.out.print("How many products will be sold? ");
        int productCount = in.nextInt();
        VendingMachine machine = new VendingMachine(product, productCount);
        System.out.print("How many coins will you enter? ");
        int coinCount = in.nextInt();
        for( int i=0; i<coinCount; i++ ) {
            System.out.print("Enter coin worth: ");
            double coin = in.nextDouble();
            if( machine.enterCoin(coin) )
                System.out.println("Coin #" + (i+1) + " accepted.");
            else
                System.out.println("Coin #" + (i+1) + " NOT accepted.");
        }
        System.out.println("You have received " + machine.buyProduct() + " items.");
        in.close();
    }
}
```

Question 4: Draw the sequence diagram of the main method you have coded in question 3. (20p)