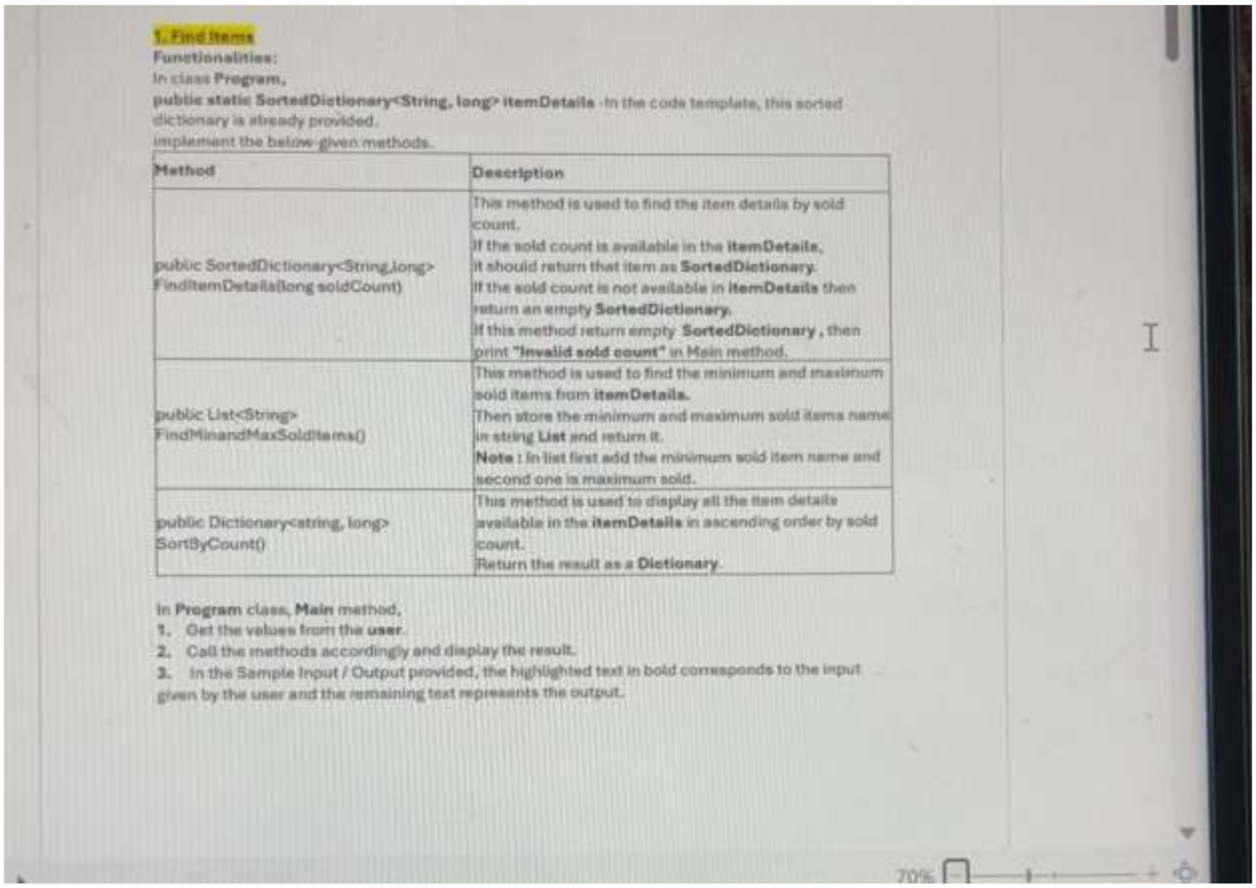


1. Find Items

Functionalities:

In class **Program,**

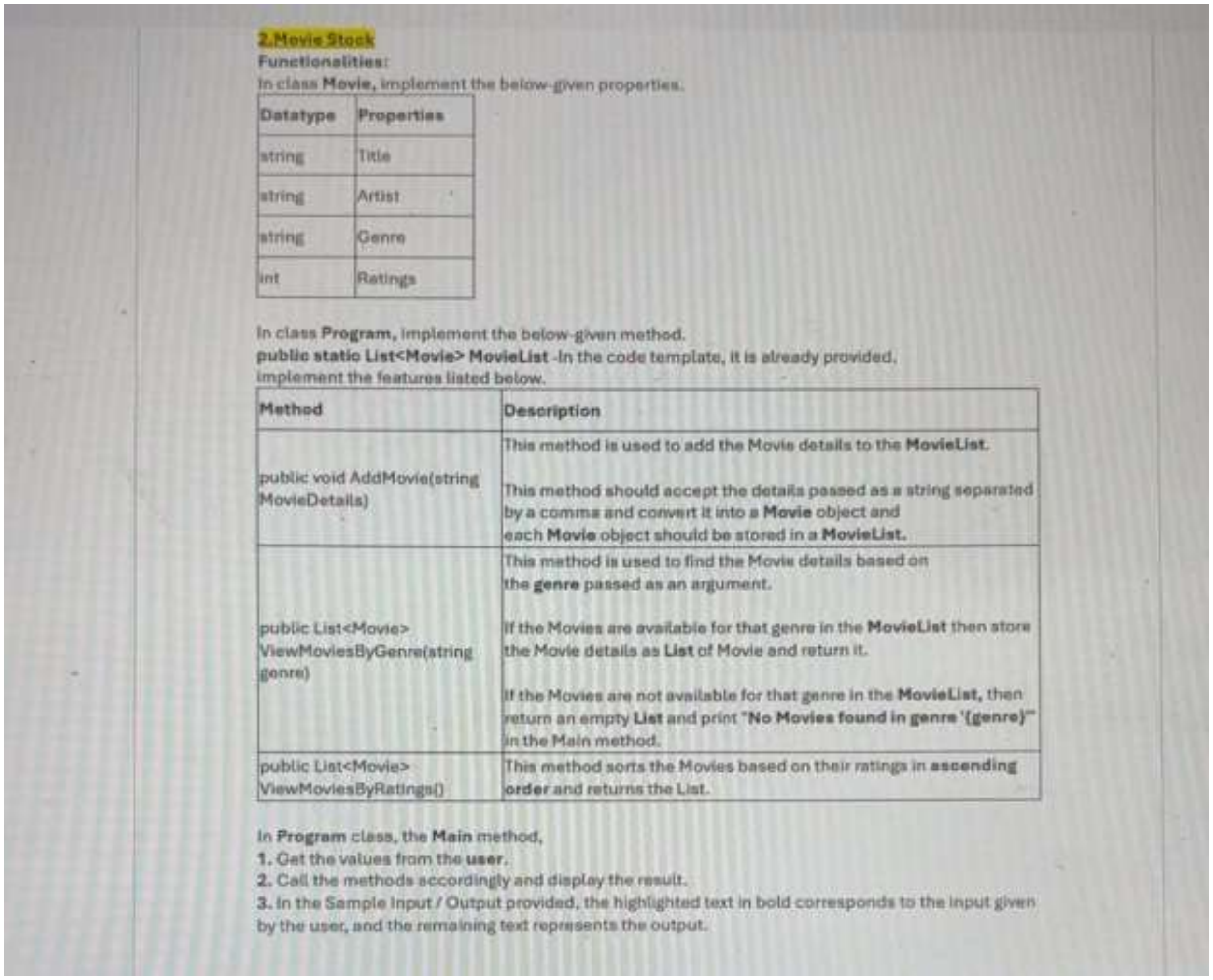
public static SortedDictionary<String, long> itemDetails -In the code template, this sorted dictionary is already provided.

implement the below-given methods.

Method	**Description**
public SortedDictionary<String,long> FindItemDetails(long soldCount)	This method is used to find the item details by sold count. If the sold count is available in the **itemDetails**, it should return that item as **SortedDictionary**. If the sold count is not available in **itemDetails** then return an empty **SortedDictionary.** If this method return empty **SortedDictionary** , then print **"Invalid sold count"** in Main method.
public List<String> FindMinandMaxSoldItems()	This method is used to find the minimum and maximum sold items from **itemDetails.** Then store the minimum and maximum sold items name in string **List** and return it. **Note :** In list first add the minimum sold item name and second one is maximum sold.
public Dictionary<string, long> SortByCount()	This method is used to display all the item details available in the **itemDetails** in ascending order by sold count. Return the result as a **Dictionary**.

In **Program** class, **Main** method,

1. Get the values from the **user**.
2. Call the methods accordingly and display the result.
3. In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.



2.Movie Stock

Functionalities:

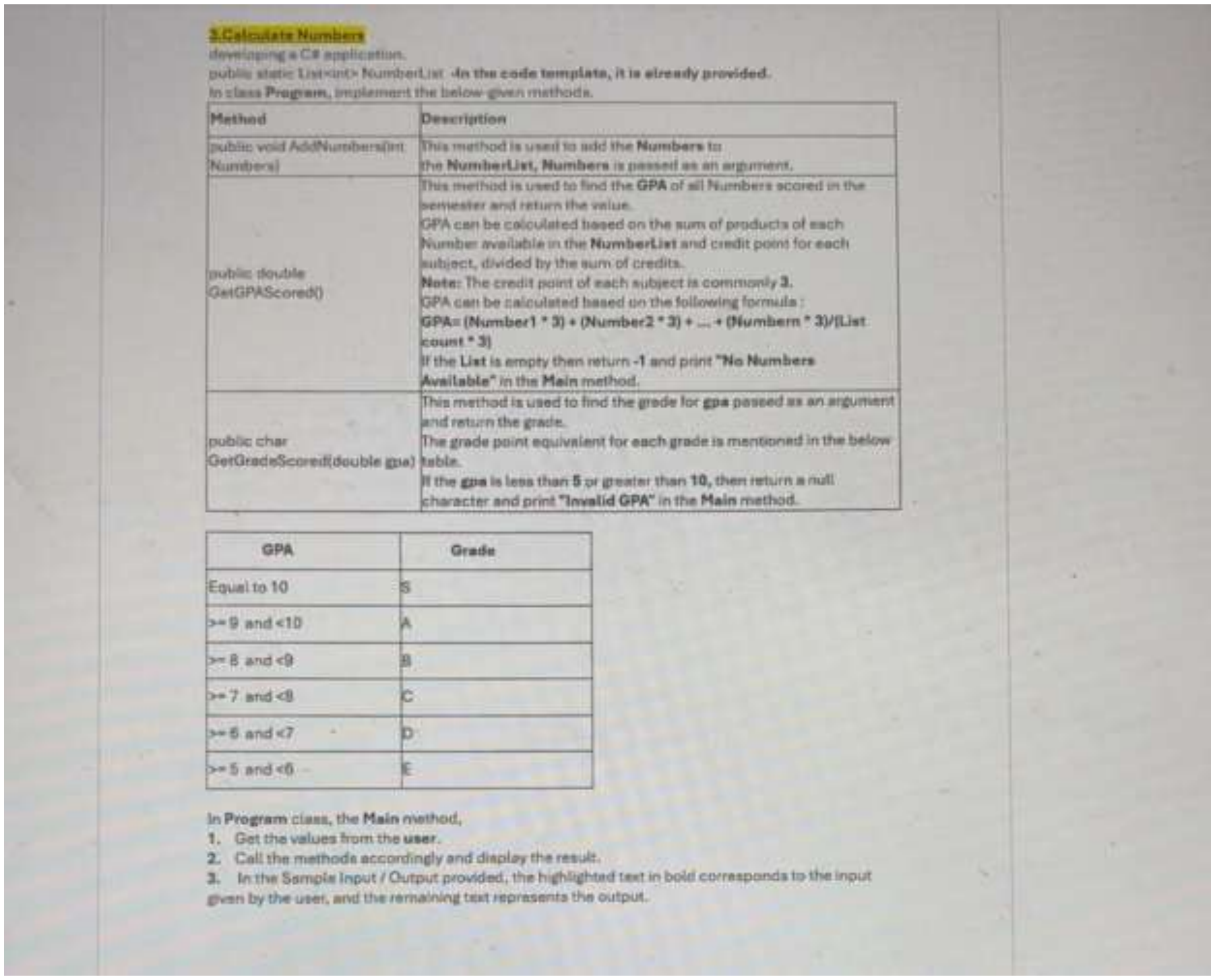
In class **Movie**, implement the below-given properties.

Datatype	Properties
string	Title
string	Artist
string	Genre
int	Ratings

In class **Program**, implement the below-given method.
public static List<Movie> MovieList -In the code template, it is already provided.
implement the features listed below.

Method	Description
public void AddMovie(string MovieDetails)	This method is used to add the Movie details to the **MovieList**. This method should accept the details passed as a string separated by a comma and convert it into a **Movie** object and each **Movie** object should be stored in a **MovieList.**
public List<Movie> ViewMoviesByGenre(string genre)	This method is used to find the Movie details based on the **genre** passed as an argument. If the Movies are available for that genre in the **MovieList** then store the Movie details as **List** of Movie and return it. If the Movies are not available for that genre in the **MovieList,** then return an empty **List** and print **"No Movies found in genre '{genre}'"** in the Main method.
public List<Movie> ViewMoviesByRatings()	This method sorts the Movies based on their ratings in **ascending order** and returns the List.

In **Program** class, the **Main** method,
1. Get the values from the **user**.
2. Call the methods accordingly and display the result.
3. In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user, and the remaining text represents the output.



3.Calculate Numbers

developing a C# application.

public static List<int> NumberList -**In the code template, it is already provided.**

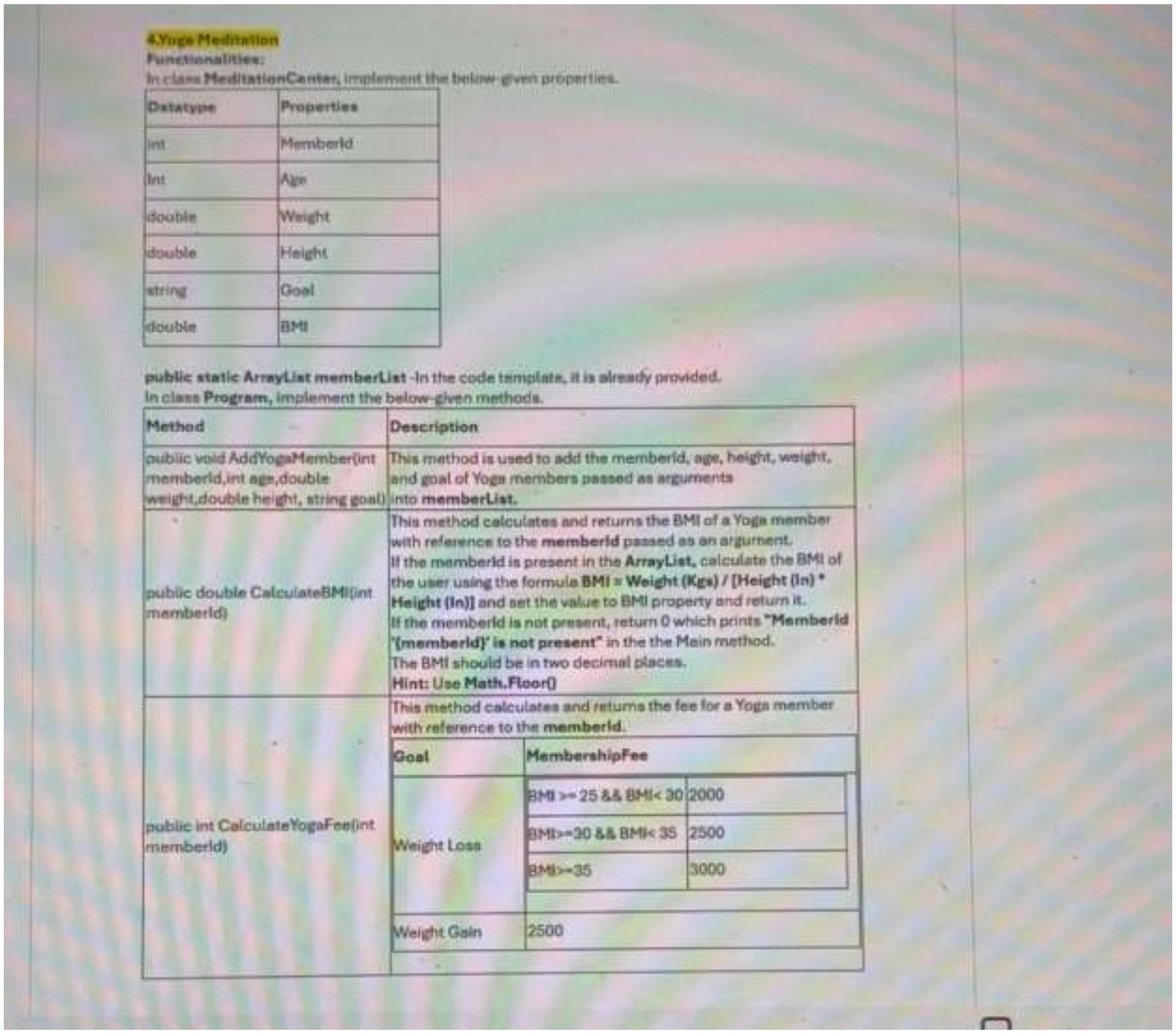
In class **Program**, implement the below-given methods.

Method	Description
public void AddNumbers(int Numbers)	This method is used to add the **Numbers** to the **NumberList, Numbers** is passed as an argument.
public double GetGPAScored()	This method is used to find the **GPA** of all Numbers scored in the semester and return the value. GPA can be calculated based on the sum of products of each Number available in the **NumberList** and credit point for each subject, divided by the sum of credits. **Note:** The credit point of each subject is commonly **3.** GPA can be calculated based on the following formula : **GPA= (Number1 * 3) + (Number2 * 3) + ... + (Numbern * 3)/(List count * 3)** If the **List** is empty then return **-1** and print **"No Numbers Available"** in the **Main** method.
public char GetGradeScored(double gpa)	This method is used to find the grade for **gpa** passed as an argument and return the grade. The grade point equivalent for each grade is mentioned in the below table. If the **gpa** is less than **5** or greater than **10,** then return a null character and print **"Invalid GPA"** in the **Main** method.

GPA	Grade
Equal to 10	S
>=9 and <10	A
>=8 and <9	B
>=7 and <8	C
>=6 and <7	D
>=5 and <6	E

In **Program** class, the **Main** method,

1. Get the values from the **user**.
2. Call the methods accordingly and display the result.
3. In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user, and the remaining text represents the output.



4.Yoga Meditation

Functionalities:

In class **MeditationCenter**, implement the below-given properties.

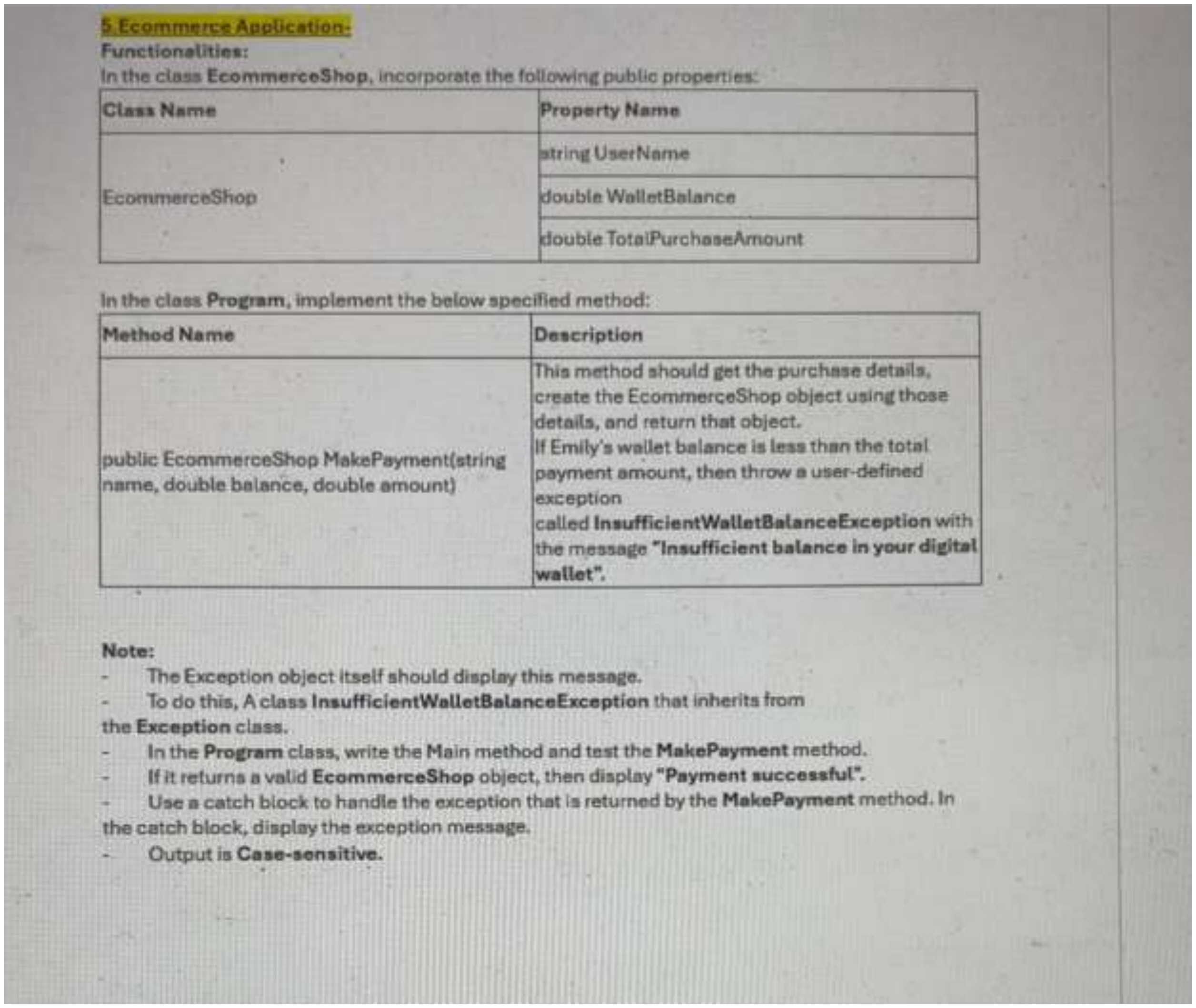
Datatype	Properties
int	MemberId
Int	Age
double	Weight
double	Height
string	Goal
double	BMI

public static ArrayList memberList -In the code template, it is already provided.

In class **Program,** implement the below-given methods.

Method	Description
public void AddYogaMember(int memberId,int age,double weight,double height, string goal)	This method is used to add the memberId, age, height, weight, and goal of Yoga members passed as arguments into **memberList.**
public double CalculateBMI(int memberId)	This method calculates and returns the BMI of a Yoga member with reference to the **memberId** passed as an argument. If the memberId is present in the **ArrayList,** calculate the BMI of the user using the formula **BMI = Weight (Kgs) / [Height (In) * Height (In)]** and set the value to BMI property and return it. If the memberId is not present, return 0 which prints **"MemberId '{memberId}' is not present"** in the the Main method. The BMI should be in two decimal places. **Hint:** Use **Math.Floor()**
public int CalculateYogaFee(int memberId)	This method calculates and returns the fee for a Yoga member with reference to the **memberId.**

Goal	MembershipFee	
Weight Loss	BMI >= 25 && BMI< 30	2000
	BMI>=30 && BMI< 35	2500
	BMI>=35	3000
Weight Gain	2500	



5.Ecommerce Application-

Functionalities:

In the class **EcommerceShop**, incorporate the following public properties:

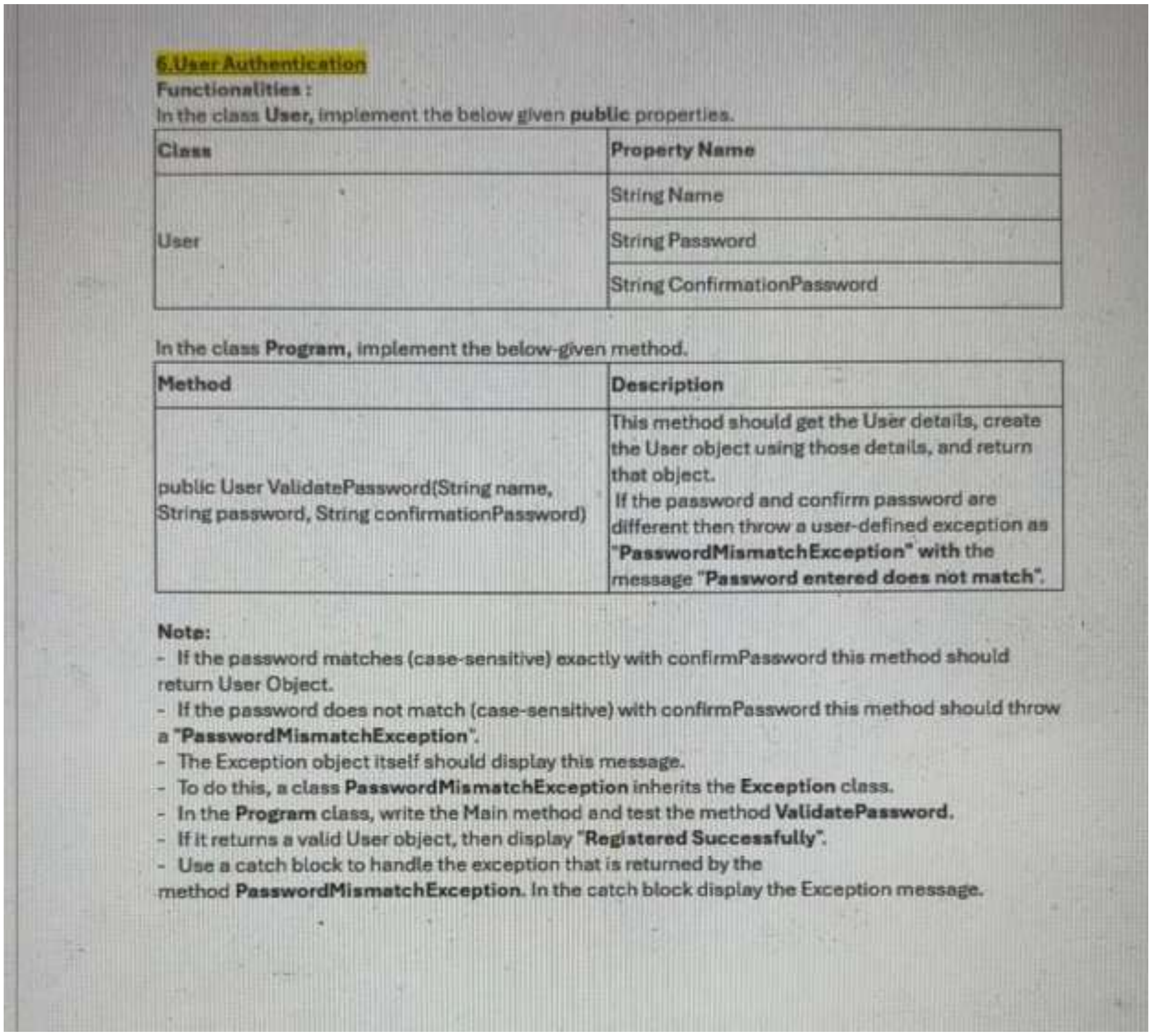
Class Name	**Property Name**
EcommerceShop	string UserName
	double WalletBalance
	double TotalPurchaseAmount

In the class **Program**, implement the below specified method:

Method Name	**Description**
public EcommerceShop MakePayment(string name, double balance, double amount)	This method should get the purchase details, create the EcommerceShop object using those details, and return that object. If Emily's wallet balance is less than the total payment amount, then throw a user-defined exception called **InsufficientWalletBalanceException** with the message **"Insufficient balance in your digital wallet".**

Note:

- The Exception object itself should display this message.
- To do this, A class **InsufficientWalletBalanceException** that inherits from the **Exception** class.
- In the **Program** class, write the Main method and test the **MakePayment** method.
- If it returns a valid **EcommerceShop** object, then display **"Payment successful".**
- Use a catch block to handle the exception that is returned by the **MakePayment** method. In the catch block, display the exception message.
- Output is **Case-sensitive.**



6.User Authentication

Functionalities :

In the class **User,** implement the below given **public** properties.

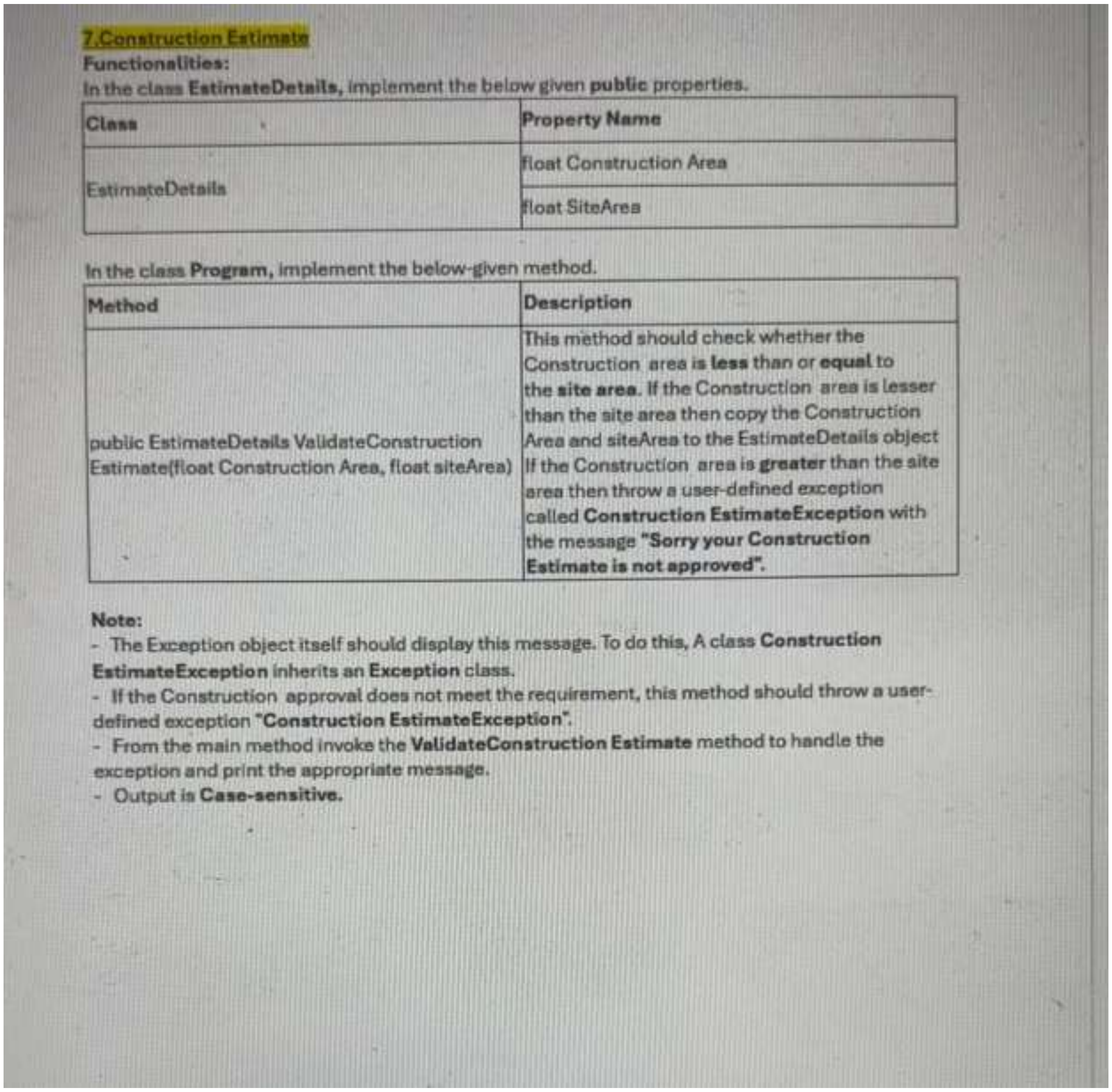
Class	**Property Name**
User	String Name
	String Password
	String ConfirmationPassword

In the class **Program,** implement the below-given method.

Method	**Description**
public User ValidatePassword(String name, String password, String confirmationPassword)	This method should get the User details, create the User object using those details, and return that object. If the password and confirm password are different then throw a user-defined exception as "**PasswordMismatchException" with** the message "**Password entered does not match**".

Note:

- If the password matches (case-sensitive) exactly with confirmPassword this method should return User Object.
- If the password does not match (case-sensitive) with confirmPassword this method should throw a "**PasswordMismatchException**".
- The Exception object itself should display this message.
- To do this, a class **PasswordMismatchException** inherits the **Exception** class.
- In the **Program** class, write the Main method and test the method **ValidatePassword.**
- If it returns a valid User object, then display "**Registered Successfully**".
- Use a catch block to handle the exception that is returned by the method **PasswordMismatchException.** In the catch block display the Exception message.



7.Construction Estimate

Functionalities:

In the class **EstimateDetails,** implement the below given **public** properties.

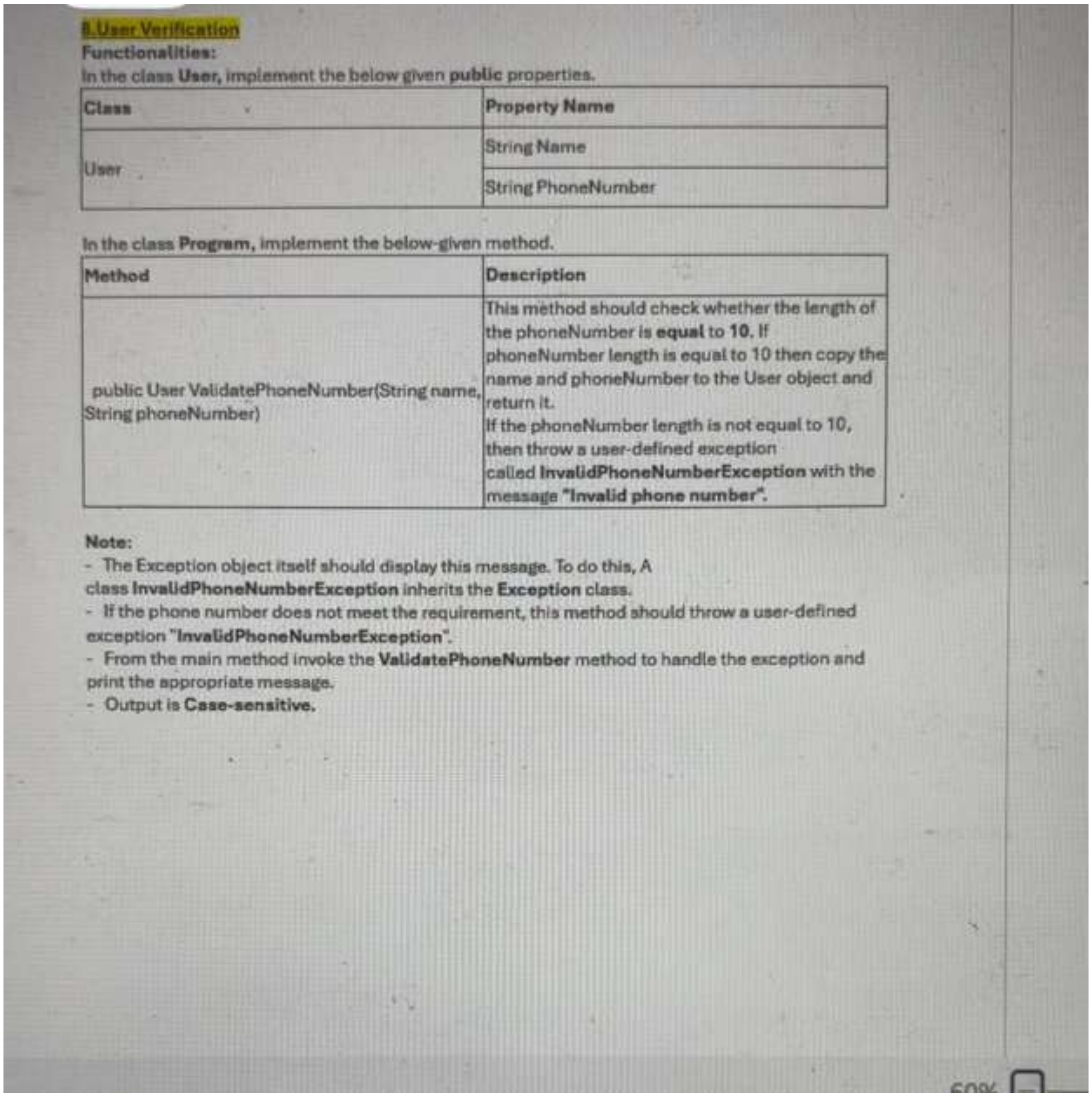
Class	**Property Name**
EstimateDetails	float Construction Area
	float SiteArea

In the class **Program,** implement the below-given method.

Method	**Description**
public EstimateDetails ValidateConstruction Estimate(float Construction Area, float siteArea)	This method should check whether the Construction area is **less** than or **equal** to the **site area.** If the Construction area is lesser than the site area then copy the Construction Area and siteArea to the EstimateDetails object If the Construction area is **greater** than the site area then throw a user-defined exception called **Construction EstimateException** with the message **"Sorry your Construction Estimate is not approved".**

Note:

- The Exception object itself should display this message. To do this, A class **Construction EstimateException** inherits an **Exception** class.
- If the Construction approval does not meet the requirement, this method should throw a user-defined exception **"Construction EstimateException".**
- From the main method invoke the **ValidateConstruction Estimate** method to handle the exception and print the appropriate message.
- Output is **Case-sensitive.**



8.User Verification

Functionalities:

In the class **User,** implement the below given **public** properties.

Class	**Property Name**
User	String Name
	String PhoneNumber

In the class **Program,** implement the below-given method.

Method	**Description**
public User ValidatePhoneNumber(String name, String phoneNumber)	This method should check whether the length of the phoneNumber is **equal** to **10.** If phoneNumber length is equal to 10 then copy the name and phoneNumber to the User object and return it. If the phoneNumber length is not equal to 10, then throw a user-defined exception called **InvalidPhoneNumberException** with the message **"Invalid phone number".**

Note:

- The Exception object itself should display this message. To do this, A class **InvalidPhoneNumberException** inherits the **Exception** class.
- If the phone number does not meet the requirement, this method should throw a user-defined exception "**InvalidPhoneNumberException**".
- From the main method invoke the **ValidatePhoneNumber** method to handle the exception and print the appropriate message.
- Output is **Case-sensitive.**