2h 39m left

Help

All

1

2

3

4

5

# 1. Person Details

Coding REJECTED

Your task here is to implement a C# code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

**Specifications:**

```
class definitions:
  class Person:
    method definitons:
        Name: Implement getter setter method (use
Auto Implementation Property)
         visibility: public
         return type: string
        Address: Implement getter setter method
(use Auto Implementation Property)
         visibility: public
         return type: string
        Age: Implement getter setter method (use
Auto Implementation Property)
         visibility: public
         return type: int
  class PersonImplementation:
        GetName(IList<Person> person): display the
name and address of all the persons in the List
         return type: String
         visibility: public

        Average(IList<Person> person): Method to
```

2h 39m left

Help

All

1

2

3

4

5

```
Average(IList<Person> person): Method to
calculate the average score
        return type: double
        visibility: public

      Max(IList<Person> person): Method to
calculate the maximum Age
        return type: int
        visibility: public
```

**Task:**

- Create a class **Person** with attributes string **Name** , string **Address** and int **age** and define getter setter method using **Auto Implementation Property**
- Create a class **PersonImplementation** and implement the below given methods:

1. **GetName(IList<Person> person)** display the name and address of all the persons in the List
2. **Average(IList<Player> players)** to calculate the *average **age*** of all the persons.
3. **Max(IList<Player> players)** to find the *maximum age* of the person.

**Sample Input**

```
IList<Person> p = new List<Person>();
      p.Add(new Person {Name = "Aarya", Address =
"A2101", Age = 69});
      p.Add(new Person {Name = "Daniel", Address =
```

2h 39m left

Help

All

1

2

3

4

5

the person.

**Sample Input**

```
IList<Person> p = new List<Person>();
      p.Add(new Person {Name = "Aarya", Address = "A2101", Age = 69});
      p.Add(new Person {Name = "Daniel", Address = "D104", Age = 40});
      p.Add(new Person {Name = "Ira", Address = "H801", Age = 25});
      p.Add(new Person {Name = "Jennifer", Address = "I1704", Age = 33});
```

**Sample Output**

```
Aarya A2101 Daniel D104 Ira H801 Jennifer I1704
41.75
69
```

**IMPORTANT:**

- If you want to test your program you can implement a **Main()** method given in the stub and you can use **RUN CODE** to test your Main(), provided you have made valid function calls with valid data required.

## Execution time limit

5 seconds

Having an issue with this question? Report

2h 39m left

Coding ACCEPTED

***Method Overloading*** is the common way of implementing polymorphism. C# can distinguish the methods with **different method signatures**. i.e. the methods can have the same name but with different parameters list (i.e. the number of the parameters, order of the parameters, and data types of the parameters) within the same class.

Your task here is to implement a **C#** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

**Specifications:**

```
class definitions:
  class Source:
      method definitions:
            Add(int a, int b, int c) : Method to
add three integer values
                return type: int
                visibility: public
           Add(double a, double b, double c) :
Method to add three double values
                return type: double
                visibility: public
```

**Task:**

Create a class **Source**, your task here is to demonstrate the function overloading by changing the data types of the parameters.

Help

All

1
2
3
4
5

2h 39m left

Help

All

1

2

3

4

5

```
        method definitions:
                Add(int a, int b, int c) : Method to
add three integer values
                    return type: int
                    visibility: public
                Add(double a, double b, double c) :
Method to add three double values
                    return type: double
                    visibility: public
```

**Task:**

Create a class **Source**, your task here is to demonstrate the function overloading by changing the data types of the parameters.

Implement the below given methods:

- **Add(int a, int b, int c) :** Method to add three integer values
- **Add(double a, double b, double c) :** Method to add three double values

**IMPORTANT:**

- If you want to test your program you can implement a **Main()** method given in the stub and you can use **RUN CODE** to test your Main(), provided you have made valid function calls with valid data required.

## Execution time limit

5 seconds

Having an issue with this question? Report

2h 39m left

Coding ATTEMPTED

Help

All

1
2
3
4
5

## Prepare Bill:

This problem is create the bill and find the total amount of the bill. Your task here is to implement a **C#** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider **default visibility** of classes, data fields and methods are public unless mentioned otherwise.

## Specification:

```
class definition:
    enum CommodityCategory: Furniture, Grocery,
Service

class Commodity:
    visibility : default
    constructor :  Commodity(CommodityCategory
category, string commodityName, int
commodityQuantity, double commodityPrice)
    visibility : public
    member property :  Category
    return type : CommodityCategory
    visibility : public
    member property :  CommodityName
    return type : string
    visibility : public
    member property :  CommodityQuantity
    return type : int
    visibility : public
    member property :  CommodityPrice
```

2h 39m left

Help

All

1

2

3

4

5

```
    member property :  CommodityPrice
    return type : double
    visibility : public
class PrepareBill:
    visibility : default
    member variable : _taxRates
    type : readonly
    return type : IDictionary<CommodityCategory,
double>
    visibility : private
    constructor : PrepareBill()
    visibility : public
    member definition :
SetTaxRates(CommodityCategory category, double
taxRate)
    return type : void
    visibility : public
    member definition :
CalculateBillAmount(IList<Commodity> items)
    visibility : public
    return type : double
```

**Task:**

**enum CommodityCategory:**

- Define the enum as mentioned in specification.

**class Commodity:**

- Define the class as mentioned in specification.

**class PrepareBill:**

2h 39m left

Help

All

1

2

3

4

5

- **PrepareBill():** Initializes _taxRates with empty dictionary.
- **SetTaxRates(CommodityCategory category, double taxRate):** Accepts commodity category and tax of that category and then its adds to _taxRates. If the tax is already added to CommodityCategory then it should not add to _taxRates.
- **CalculateBillAmount(IList<Commodity> items):** It accepts a list of commodities and it should return the total amount of the bill. If the commodity category is not present in the _taxRates, then it should throw ArgumentException.

**Sample input:**

```
var commodities = new List<Commodity>()
      {
        new Commodity(CommodityCategory.Furniture,
"Bed", 2, 50000),
        new Commodity(CommodityCategory.Grocery,
"Flour", 5, 80),
        new Commodity(CommodityCategory.Service,
"Insurance", 8, 8500)
      };
      var prepareBill = new PrepareBill();
      prepareBill.SetTaxRates(CommodityCategory.Furni
18);
      prepareBill.SetTaxRates(CommodityCategory.Groce
5);
      prepareBill.SetTaxRates(CommodityCategory.Servi
12);
```

2h 39m left

Help

All

1
2
3
4
5

```
"Flour", 5, 80),
        new Commodity(CommodityCategory.Service,
"Insurance", 8, 8500)
    };
    var prepareBill = new PrepareBill();
    prepareBill.SetTaxRates(CommodityCategory.Furni
18);
    prepareBill.SetTaxRates(CommodityCategory.Groce
5);
    prepareBill.SetTaxRates(CommodityCategory.Servi
12);

    var billAmount =
prepareBill.CalculateBillAmount(commodities);
    Console.WriteLine($"{billAmount}");
```

**Sample output:**

```
194580
```

**Note:**

- You can make suitable function calls and use **the RUN CODE** button to check your **main()** method output.

## Execution time limit

2 seconds

Having an issue with this question? Report

2h 38m left

Help

All

1 2 3 4 5

Coding ATTEMPTED

## Broadband Plans:

This problem is related to implementing the functionality of getting the broadband subscription plan based as per the criteria mentioned below.

Your task here is to implement a **C#** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider **default visibility** of classes, data fields and methods are public unless mentioned otherwise.

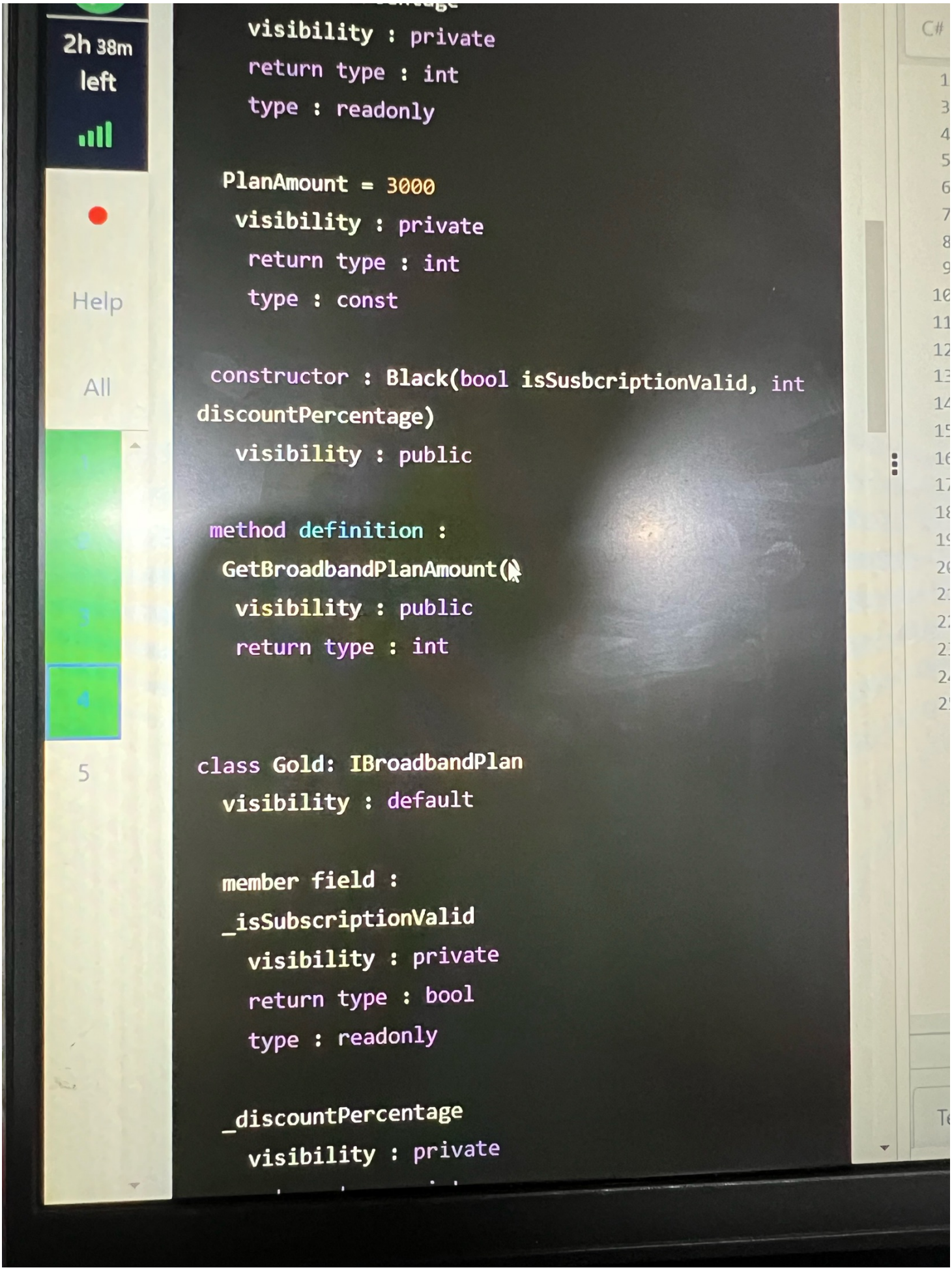
## Specifications:

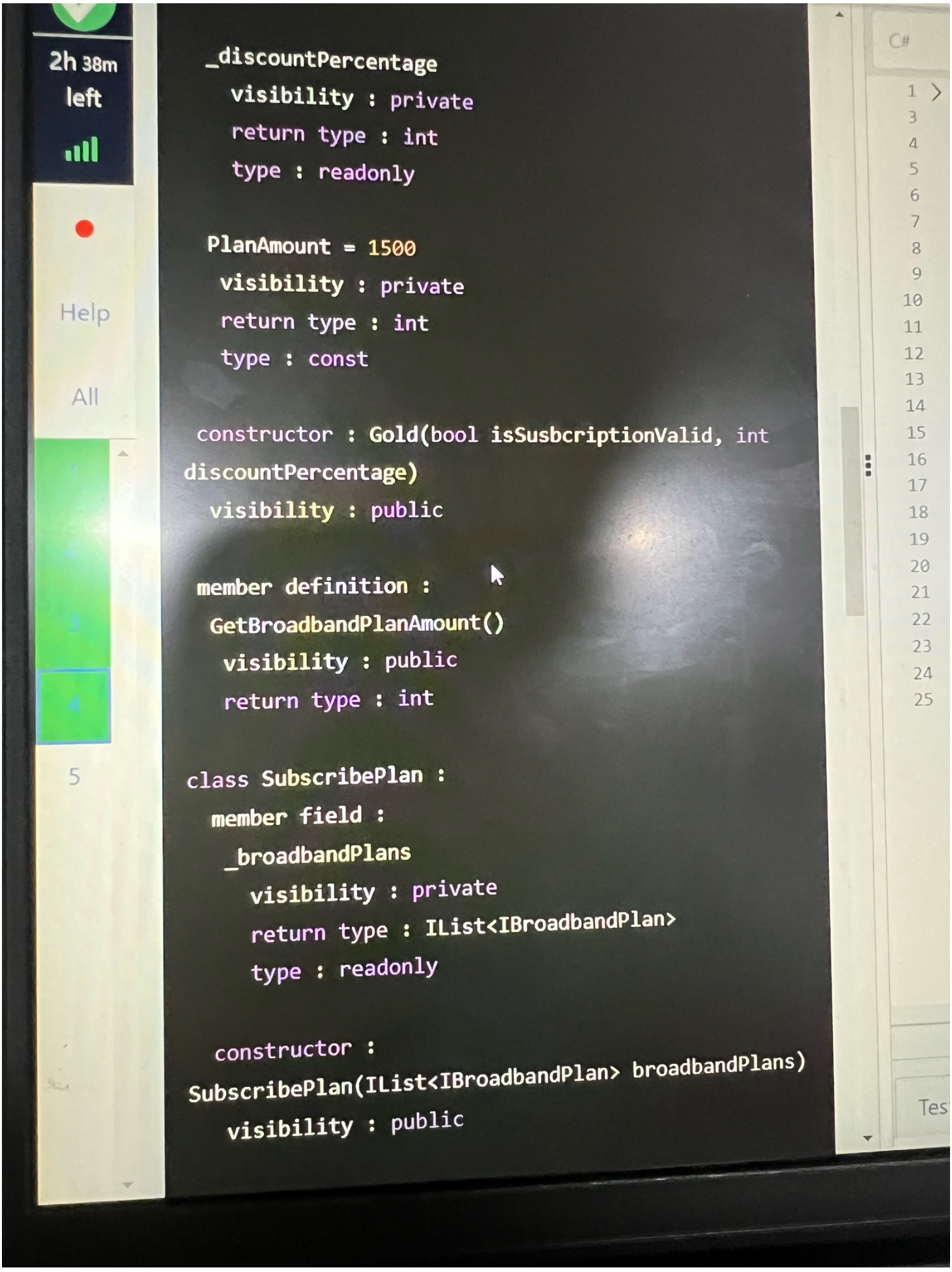
```
class definition:
 interface IBroadbandPlan :
  visibility : default

 member definition :
  GetBroadbandPlanAmount();
   return type : int
   visibility : default

class Black : IBroadbandPlan
  visibility : default

 member field:
  _isSubscriptionValid
    visibility : private
     return type : bool
     type : readonly
```


2h 38m left
Help
All
visibility : private
return type : int
type : readonly
PlanAmount = 3000
visibility : private
return type : int
type : const
constructor : Black(bool isSusbcriptionValid, int discountPercentage)
visibility : public
method definition :
GetBroadbandPlanAmount()
visibility : public
return type : int
class Gold: IBroadbandPlan
visibility : default
member field :
_isSubscriptionValid
visibility : private
return type : bool
type : readonly
_discountPercentage
visibility : private


2h 38m left
Help
All
1
2
3
4
5
_discountPercentage
visibility : private
return type : int
type : readonly
PlanAmount = 1500
visibility : private
return type : int
type : const
constructor : Gold(bool isSusbcriptionValid, int discountPercentage)
visibility : public
member definition :
GetBroadbandPlanAmount()
visibility : public
return type : int
class SubscribePlan :
member field :
_broadbandPlans
visibility : private
return type : IList<IBroadbandPlan>
type : readonly
constructor : SubscribePlan(IList<IBroadbandPlan> broadbandPlans)
visibility : public
C#
1
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

2h 38m left

Help

All

1

2

3

4

5

```
  constructor :
SubscribePlan(IList<IBroadbandPlan> broadbandPlans)
   visibility : public

   member definition :
    GetSubscriptionPlan()
     visibility : public
     return type : IList<Tuple<string, int>>
```

**Task:**

**class Black:**

- **Black(bool isSusbcriptionValid, int discountPercentage):** Accepts two parameters and initializes _isSubscriptionValid and _discountPercentage. It should throw ArgumentOutOfRangeException if discountPercentage is less than 0 and greater than 50.
- **int GetBroadbandPlanAmount():** If the subscription is valid then it should return the discounted price otherwise it should return the normal price.

**class Gold:**

- **Gold(bool isSusbcriptionValid, int discountPercentage):** Accepts two parameters and initializes _isSubscriptionValid and _discountPercentage. It should throw ArgumentOutOfRangeException if discountPercentage is less than 0 and greater than 30.
- **int GetBroadbandPlanAmount():** If the subscription is valid

C#

1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

2h 38m left

Help

All

1
2
3
4
5

**isSubscriptionValid, int discountPercentage):** Accepts two parameters and initializes _isSubscriptionValid and _discountPercentage. It should throw ArgumentOutOfRangeException if discountPercentage is less than 0 and greater than 30.

- **int GetBroadbandPlanAmount():** If the subscription is valid then it should return the discounted price otherwise it should return the normal price.

**class SubscribePlan:**

- **SubscribePlan(IList<IBroadbandPlan> broadbandPlans):** Accepts a list of Broadband Plans and initializes the _customers field. If the input null than it should throw ArgumentNullException.
- **IList<Tuple<string, int>> GetSubscriptionPlan():** It should return a list which contains plan type and its subscription plan amount. If _broadbandPlans is null than it should throw ArgumentNullException.

**Sample input:**

```
var plans = new List<IBroadbandPlan>
  {
    new Black(true, 50),
    new Black(false, 10),
    ,
    ),
    new Gold(false, 20)
  };
  var subscriptionPlans = new
```

Slow Internet Connection

C#

1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Tes

```
            new Black(true, 50),
            new Black(false, 10),
            new Gold(true, 30),
            new Black(true, 20),
            new Gold(false, 20)
        };
        var subscriptionPlans = new
SubscribePlan(plans);
        var result =
subscriptionPlans.GetSubscriptionPlan();
        foreach (var item in result)
        {
            Console.WriteLine($"{item.Item1},
{item.Item2}");
        }
```

**Sample output:**

```
Black, 1500
Black, 3000
Gold, 1050
Black, 2400
Gold, 1500
```

**Note:**

- You can make suitable function calls and use **the RUN CODE** button to check your **main()** method output.

## Execution time limit

2 seconds

2h 38m left

Help

All

1
2
3
4
5

# 5. Student Scholarship

Coding | UNSOLVED

## Problem Description

Your task here is to implement a **C#** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

## Specifications

```
Delegate definations:
  IsEligibleforScholarship(Student std)
  return type: bool
  visibility: public

Class Student:
  Auto implemented Properties
   RollNo : int
    visibility: public
   Name: string
    visibility: public
   Marks: int
    visibility: public
  SportsGrade: char
    visibility: public

  Method definitions:
   static GetEligibleStudents(List<Student>
studentsList, IsEligibleforScholarship isEligible)
     return type: string
     visibility: public
```

C#

1
2
3
4
5
6
7
8
9
10
11

Test

2h 38m left

Help

All

```
...ligibleStudents(List<Student>
studentsList, IsEligibleforScholarship isEligible)
    return type: string
    visibility: public

Class Program
  Method definitions:
   static ScholarshipEligibility(Student std)
    return type : bool
    visibility: public
```

## Task

- Create **Program** class and implement below given method
- static **ScholarshipEligibility(Student std)** - Return whether student is eligible for scholarship.
- Create **Student** Class and implement below given method
- static **GetEligibleStudents(List<Student> studentsList, IsEligibleforScholarship isEligible)** - returns comma separated list of eligible students.
- Create **delegate** as below
- **IsEligibleforScholarship(Student student)** - return true or false.
- Below are criteria for student to eligible for scholarship

1. Marks of student should be more than 80.
2. SportsGrade of student should be 'A'.
3. Pass list of students and call delegate to check eligibility.

## Sample Input

C#

1 2 3 4 5 6 7 8 9 10 11

1 2 3 4 5

Tes

2h 38m left

Help

All

1. Marks of student should be more than 80.
2. SportsGrade of student should be 'A'.
3. Pass list of students and call delegate to check eligibility.

## Sample Input

```
    List<Student> lstStudents = new List<Student>
();
    Student obj1 = new Student() { RollNo = 1,
Name = "Raj", Marks = 75, SportsGrade = 'A' };
    Student obj2 = new Student() { RollNo = 2,
Name = "Rahul", Marks = 82, SportsGrade = 'A' };
    Student obj3 = new Student() { RollNo = 3,
Name = "Kiran", Marks = 89, SportsGrade = 'B' };
    Student obj4 = new Student() { RollNo = 4,
Name = "Sunil", Marks = 86, SportsGrade = 'A' };
```

## Sample Output

```
Rahul, Sunil
```

## IMPORTANT

- If you want to test your program you can implement a **Main()** method and you can use **RUN CODE** to test your Main(), provided you have made valid function calls with valid data required.

## Execution time limit

5 seconds

Having an issue with this question? Report

```
7       public int Age { get;set; }
8   }
9
10  public class PersonImplementation
11  {
12      public string GetName(IList<Person> persons)
13      {
14          string result = "";
15          foreach(var i in persons)
16          {
17              result += $"{i.Name} {i.Address} ";
18          }
19          return result;
20      }
21
22      public double Average(IList<Person> persons)
23      {
24          double sum = 0;
25          double count = 0;
26          foreach(var i in persons)
27          {
28              sum += i.Age;
29              count++;
30          }
31          return sum/count;
32      }
33
34      public int Max(IList<Person> persons)
35      {
36          int max =0;
37          foreach(var i in persons){
38              if(i.Age>max)
39              max = i.Age;
40          }
41          return max;
42      }
43  }
```

Test Results | Custom Input

ogle recommends setting Chrome as default **Set as default**

- Create a class **Person** with attributes string **Name** , string **Address** and int **age** and define getter setter method using **Auto Implementation Property**
- Create a class **PersonImplementation** and implement the below given methods:

1. **GetName(IList<Person> person)** display the name and address of all the persons in the List
2. **Average(IList<Player> players)** to calculate the *average* ***age*** of all the persons.
3. **Max(IList<Player> players)** to find the *maximum age* of the person.

**Sample Input**

```
IList<Person> p = new List<Person>();
    p.Add(new Person {Name = "Aarya", Address = "A2101", Age = 69});
    p.Add(new Person {Name = "Daniel", Address = "D104", Age = 40});
    p.Add(new Person {Name = "Ira", Address = "H801", Age = 25});
    p.Add(new Person {Name = "Jennifer", Address = "I1704", Age = 33});
```

**Sample Output**

```
Aarya A2101 Daniel D104 Ira H801 Jennifer I1704
41.75
69
```

**IMPORTANT:**

- If you want to test your program you can implement a **Main()** method given in the stub and you can use **RUN CODE** to test your Main(), provided you have made valid function calls with valid data required.

Execution time limit

5 seconds

Having an issue with this question? Report