Universal Home Appliances

1 Hrs 57 Mins 1 Secs

Assessment Home

Scenario:

Universal Home Appliances is a leading manufacturer and distributor of top-quality household appliances, including refrigerators, ovens, dishwashers, and washing machines. The company is committed to enhancing their supply chain and logistics operations to guarantee timely delivery of products to retailers and customers. To further improve customer satisfaction, the company is seeking to implement software to streamline the purchasing process, schedule repairs, and manage appliance information more efficiently.

You being their software consultant, help them to do this task by developing a C# application

Functionalities:

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

Data Type	Property Name
string	Id
string	Name
string	Brand

Functionalities:

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

Data Type	Property Name
string	Id
string	Name
string	Brand
double	Price

In class **Program,**

public static Dictionary<int, Appliance> applianceDetails -This is already given in the code template. Values for this Dictionary is also provided.

Implement the below-given methods.

Method	Description
public Dictionary<string, string>	This method is used to get the appliance details

In class **Program,**

public static Dictionary<int, Appliance> applianceDetails -This is already given in the code template. Values for this Dictionary is also provided.

Implement the below-given methods.

Method	Description
public Dictionary<string, string> GetApplianceDetails(string applianceId)	This method is used to get the appliance details by **Id**. If the appliance id is available in the **applianceDetails** Dictionary, then store the appliance id and name as **Dictionary** and return it. **Note:** Before adding the name to the dictionary, join the brand with an underscore (_). **For Example** : Appliance name : Oven Brand : LG then add a name to the dictionary, like **Oven_LG.**

public Dictionary<string, string> FindApplianceWithPriceRange(double minRange,double maxRange)	This method is used to find the appliance details based on the minimum and maximum price range from **applianceDetails**. **Note :** Maximum and Minimum Range **(Both inclusive)** If the appliance is available within the given price range, then store the appliance name and brand to the **Dictionary** and return it. If the appliance is not available within the given price range, then return an **empty Dictionary** with count 0.
public Dictionary<int, Appliance> FindHighCostAppliance()	This method is used to find the high cost appliance details from the **applianceDetails** Dictionary. Return the result as a **Dictionary**.

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

1. Get the values from the **user**.
2. Call the methods accordingly.
3. If the **GetApplianceDetails** method returns an empty Dictionary, then display **"Appliance id not found"** in **Main** method.
4. If the **FindApplianceWithPriceRange** method returns an empty Dictionary, then display **"Appliance not found"** in Main method.
5. Display the result as per the Sample Input/Output given.

Note:

- Keep the method and class as **public.**
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- 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.
- Do not change the given code template.

Sample Input / Output:

Sample Input / Output:

1. Get appliance details

2. Find appliance with price range

3. Find high cost appliance

4. Exit

Enter your choice

1

Enter the appliance id

AP02

AP02 Dishwasher_Samsung

1. Get appliance details

2. Find appliance with price range

3. Find high cost appliance

4. Exit

AP02

AP02 Dishwasher_Samsung

1. Get appliance details

2. Find appliance with price range

3. Find high cost appliance

4. Exit

Enter your choice

2

Enter the minimum price range

1000

Enter the maximum price range

5000

Oven Bosch

Toaster LG

Enter the minimum price range

1000

Enter the maximum price range

5000

Oven Bosch

Toaster LG

1. Get appliance details

2. Find appliance with price range

3. Find high cost appliance

4. Exit

Enter your choice

2

Enter the minimum price range

500

Enter your choice

2

Enter the minimum price range

500

Enter the maximum price range

1000

Appliance not found

1. Get appliance details

2. Find appliance with price range

3. Find high cost appliance

4. Exit

Enter your choice

3

ID : AP02, Name : Dishwasher, Brand : Samsung Price : 25000

1. Get appliance details

2. Find appliance with price range

3. Find high cost appliance

4. Exit

Enter your choice

3

ID : AP02, Name : Dishwasher, Brand : Samsung Price : 25000

1. Get appliance details

2. Find appliance with price range

3. Find high cost appliance

4. Exit

Enter your choice

4

Thank You

Elegant Jewels

1 Hrs 54 Mins 12 Secs

Assessment Home

Scenario:

Elegant Jewels is a well-known and reputable jewellery store in the city. They are preparing for Valentine's Day, which is a busy time for them, and they want to increase their sales of gold and silver coins. To do this, they plan to offer pre-booking for customers to purchase these coins in advance. The management team wants to develop an application that will allow customers to pre-book their purchases of gold and silver coins and also have the ability to calculate the total price of the coins based on the current market value of gold and silver.

You being their software consultant help them by developing a C# application.

Functionalities:

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

Datatype	Property Name
string	MetalName
double	Weight

Functionalities:

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

Datatype	Property Name
string	MetalName
double	Weight
double	PurityOfMetal
bool	WantDecoration

In class **Service,** implement the below-given methods and also **Inherit** the class **Bill**.

Method	Description
public void ExtractDetails(string billDetails)	This method accepts **billDetails** as an argument and extracts the values from the argument using a **colon** and assign those values to the **Bill** class **properties**. In this method, pass the details as an argument with the following string format.

In class **Service,** implement the below-given methods and also **Inherit** the class **Bill**.

Method	Description
public void ExtractDetails(string billDetails)	This method accepts **billDetails** as an argument and extracts the values from the argument using a **colon** and assign those values to the **Bill** class **properties**. In this method, pass the details as an argument with the following string format. **<MetalName:Weight:PurityOfMetal:WantDecoration>**
public bool ValidateMetalName()	This method is used to validate the metal name of the Bill object. The metal name should be "**Gold**" or "**Silver**". If the metal name is valid then return **true**. Otherwise, return **false.** **Note:** Metal Name is **Case-sensitive**.

public bool ValidateMetalName()	This method is used to validate the metal name of the Bill object. The metal name should be "**Gold**" or "**Silver**". If the metal name is valid then return **true**. Otherwise, return **false.** **Note:** Metal Name is **Case-sensitive**.
public double CalculateTotalPrice()	This method is used to calculate the total price and return the same. The calculation procedures are given below.

Formula :

Total Price = (Price Per Gram*(Purity Of Metal/100))*Weight.

If WantDecoration is **true**, then 10% of the total price should be added.

Formula :

Total Price = (Price Per Gram*(Purity Of Metal/100))*Weight.

If WantDecoration is **true**, then 10% of the total price should be added.

Metal Name	Price per gram of 100 % purity
Gold	5000
Silver	100

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

1. Get the values from the **user** as a string.

2. Call the **ExtractDetails** method and pass that string.

3. Call the **ValidateMetalName** method, If it returns **true** then move on to step 4, If it returns **false** then display **Invalid metal name**

4. Use the values in the method **CalculateTotalPrice** and display the result as per the Sample Output.

Note:

- Keep the properties, methods and classes as **public.**
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

Sample Input 1:

Enter the bill details

Gold:2:91.6:true

Sample Output 1:

The bill amount is: 10076

Sample Input 2:

Enter the bill details

Sample Input 1:

Enter the bill details

Gold:2:91.6:true

Sample Output 1:

The bill amount is: 10076

Sample Input 2:

Enter the bill details

Copper:10:80:false

Sample Output 2:

Invalid metal name

Archive Management

1 Hrs 53 Mins 16 Secs

Assessment Home

Scenario:

An archive management system needs software that assists libraries in managing their collections, cataloguing information, tracking members, and circulating materials. The system typically includes a database of the library's inventory as well as modules for checking out and returning books, managing fines and fees, and generating reports.

As their software consultant, you can help them by developing a C# application.

Functionalities:

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

Data Type	Properties
string	MemberID
string	Name
string	Genre

Functionalities:

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

Data Type	Properties
string	MemberID
string	Name
string	Genre
string	IssueDate
string	ReturnDate
double	Penalty

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

public static SortedDictionary<int, Book> bookDetails -In the code template, it is already provided.

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

public static SortedDictionary<int, Book> bookDetails -In the code template, it is already provided.

implement the features listed below.

Method	Description
public SortedDictionary<string, List<Book>> GroupBooksByGenre()	This method is used to group books by genre. Then store the results in SortedDictionary and return that **SortedDictionary**.
public Dictionary<string,double> UpdatePenaltyAmount(double amount)	This method is used to update the penalty amount passed as an argument for member who returned the book after three days. Then store the **memberID** and the member's **Penalty** amount in **Dictionary** and return them.
public List<string> FindBooksNameWithSameDayReturn()	This method is used to find the books which is returned as the same date of issue. Then store the book's name in the **List** and return it.

public Dictionary<string,double> UpdatePenaltyAmount(double amount)	This method is used to update the penalty amount passed as an argument for member who returned the book after three days. Then store the **memberID** and the member's **Penalty** amount in **Dictionary** and return them.
public List<string> FindBooksNameWithSameDayReturn()	This method is used to find the books which is returned as the same date of issue. Then store the book's name in the **List** and return it.

Note : The date format is **(MM/dd/yyyy).**

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.

Note:

- Keep the method and class as **public.**
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

Sample Input / Output:

1. Group books by genre

2. Update penalty amount

3. Find same day return books

4. Exit

Enter your choice

1

Adventure
The Stranger Odyssey

1

Adventure
The Stranger Odyssey

Fantasy
The Hobbit The Alchemist

Historical
War and Peace

1. Group books by genre

Historical
War and Peace

1. Group books by genre

2. Update penalty amount

3. Find same day return books

4. Exit

Enter your choice

2

Enter the penalty amount

100

MemeberID	Penalty
M01	100
M04	100

2

Enter the penalty amount

100

MemeberID	Penalty
M01	100
M04	100

1. Group books by genre

2. Update penalty amount

3. Find same day return books

4. Exit

Enter your choice

3

1. Group books by genre

2. Update penalty amount

3. Find same day return books

4. Exit

Enter your choice

3

War and Peace

Odyssey

1. Group books by genre

2. Update penalty amount

3. Find same day return books

4. Exit

Enter your choice

4

1. Group books by genre

2. Update penalty amount

3. Find same day return books

4. Exit

Enter your choice

4

Thank You

Motors Hut

1 Hrs 51 Mins 47 Secs

Assessment Home

Scenario:

Motors Hut is a well-known motor parking spot in the city. They often have designated parking spots for each vehicle and may also have security or surveillance systems in place to ensure the safety of the vehicles. The management team wants to develop an application that will allow customers to reserve parking spots in advance, view available spots, and calculate the parking fee based on the type of vehicle and the duration of parking.

You being their software consultant help them by developing a C# application.

Functionalities:

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

Datatype	Property Name
string	VehicleNumber
string	VehicleType
int	NumberOfDays

Functionalities:

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

Datatype	Property Name
string	VehicleNumber
string	VehicleType
int	NumberOfDays

In class **Service,** implement the below-given methods and also **Inherit** the class **Parking**.

Method	Description
public void ExtractDetails(string parkingDetails)	This method accepts **parkingDetails** as an argument and extracts the values from the argument using a **colon (:)** and assign those values to the **Parking** class **properties**. In this method, pass the details as an argument with the

In class **Service,** implement the below-given methods and also **Inherit** the class **Parking**.

Method	Description
public void ExtractDetails(string parkingDetails)	This method accepts **parkingDetails** as an argument and extracts the values from the argument using a **colon (:)** and assign those values to the **Parking** class **properties**. In this method, pass the details as an argument with the following string format. **<VehicleNumber:VehicleType:NumberOfDays>**
public bool ValidateVehicleType()	This method is used to validate the vehicle type of Parking object. The vehicle type should be "**2 Wheeler**" or "**3 Wheeler**" or "**4 Wheeler**". If the vehicle type is valid then return **true**. Otherwise, return **false.** **Note:**

public bool ValidateVehicleType()	This method is used to validate the vehicle type of Parking object. The vehicle type should be "**2 Wheeler**" or "**3 Wheeler**" or "**4 Wheeler**". If the vehicle type is valid then return **true**. Otherwise, return **false.** **Note:** Vehicle Type is **Case-sensitive**.
public double CalculateTotalAmount()	This method is used to calculate the total amount and return the **amount**. The calculation procedures are given below.

Formula :

Total Amount = number Of Days * Price per day

Vehicle Type	Price per day

Formula :

Total Amount = number Of Days * Price per day

Vehicle Type	Price per day
2 Wheeler	50
3 Wheeler	70
4 Wheeler	100

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

1. Get the values from the **user** as a string.

2. Call the **ExtractDetails** method and pass that string.

3. Call the **ValidateVehicleType** method, If it returns **true** then move on to step 4, If it returns **false** then display **Invalid vehicle type**

4. Use the values in the method **CalculateTotalAmount** and display the result as per the Sample Output.

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

1. Get the values from the **user** as a string.

2. Call the **ExtractDetails** method and pass that string.

3. Call the **ValidateVehicleType** method, If it returns **true** then move on to step 4, If it returns **false** then display **Invalid vehicle type**

4. Use the values in the method **CalculateTotalAmount** and display the result as per the Sample Output.

Note:

- Keep the properties, methods and classes as **public.**
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

Sample Input 1:

Enter the parking details

TN72BQ6519:2 Wheeler:4

Sample Output 1:

Sample Input 1:

Enter the parking details

TN72BQ6519:2 Wheeler:4

Sample Output 1:

The amount to be paid: 200

Sample Input 2:

Enter the parking details

TN92OP0307:5 Wheeler:3

Sample Output 2:

Invalid vehicle type