**Lighting the Way**

As the keeper of the lighthouse, Jack's job was to ensure that the light was operating properly and emitting the correct number of flashes. The light was set to flash every 72 seconds, and Jack needed to calculate the number of flashes for the given hours. Jack decided to automate the process. As a software developer, help Jack calculate the number of flashes for the given hours.

**Component Specification: LightHouse**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Responsibilities** |
| **LightHouse** | String location  int height  String lightType  int hours | Necessary getters, setters and four argument constructor are provided as part of the code skeleton. |

**Functional Requirement 1: Extract the details of LightHouse and create an object of the LightHouse class**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Methods** | **Responsibilities** |
| **UseInterface** | public static lightHouse **extractDetails**(String lightHouseDetails) | This method accepts **lightHouseDetails**separated by colon as an argument and should extract the properties of the **LightHouse**from the argument by parsing the **lightHouseDetails**. Set these values to the **LightHouse**object and return this object. |

**Functional Requirement 2: Calculate flashes for a given hours and return the result.**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Methods** | **Responsibilities** |
| **LightHouse** | public int **calculateFlashesForGivenHours** () | This method is used to calculate the flashes and return the result.  Calculate flashes if the hours and heights are greater than zero.  Else return -1 and display "**Invalid light house details**".  ***Condition:***   * ***Hours should be positive****.* * ***Height should be positive****.* |

**Formula to calculate flashes for given hours**

**Flashes=( hours \* 60 \* 60 ) / 72**

For example, Let hour = 10

**Flashes**= ( 10 \* 60 \* 60 ) / 72

**Flashes**= 500

**The main method in the UserInterface class is excluded from evaluation. You are free to write your own code in the main method to invoke the business methods to check its correctness.**

**Note:**

* In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
* Ensure to follow the object-oriented specifications provided in the question
* Ensure to provide the names for classes, attributes, and methods as specified in the question.
* Adhere to the code template, if provided.

**Sample Input/Output -1**

Enter the details

**Genoa:247:SpecialBeam:10**

LightHouse Location: Genoa

Height: 247

Light Type: SpecialBeam

Hour: 10

Flashes: 500

**Sample Input/Output -2**

Enter the details

**Vasto:307:StrongIntensity:-15**

Invalid light house details

**Sample Input/Output -3**

Enter the details

**Paris:-89:StrongIntensity:15**

Invalid light house details

**Lighting the Way**

As the keeper of the lighthouse, Jack's job was to ensure that the light was operating properly and emitting the correct number of flashes. The light was set to flash every 72 seconds, and Jack needed to calculate the number of flashes for the given hours. Jack decided to automate the process. As a software developer, help Jack calculate the number of flashes for the given hours.

**Component Specification: LightHouse**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Responsibilities** |
| **LightHouse** | String location  int height  String lightType  int hours | Necessary getters, setters and four argument constructor are provided as part of the code skeleton. |

**Functional Requirement 1: Extract the details of LightHouse and create an object of the LightHouse class**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Methods** | **Responsibilities** |
| **UseInterface** | public static lightHouse **extractDetails**(String lightHouseDetails) | This method accepts **lightHouseDetails**separated by colon as an argument and should extract the properties of the **LightHouse**from the argument by parsing the **lightHouseDetails**. Set these values to the **LightHouse**object and return this object. |

**Functional Requirement 2: Calculate flashes for a given hours and return the result.**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Methods** | **Responsibilities** |
| **LightHouse** | public int **calculateFlashesForGivenHours** () | This method is used to calculate the flashes and return the result.  Calculate flashes if the hours and heights are greater than zero.  Else return -1 and display "**Invalid light house details**".  ***Condition:***   * ***Hours should be positive****.* * ***Height should be positive****.* |

**Formula to calculate flashes for given hours**

**Flashes=( hours \* 60 \* 60 ) / 72**

For example, Let hour = 10

**Flashes**= ( 10 \* 60 \* 60 ) / 72

**Flashes**= 500

**The main method in the UserInterface class is excluded from evaluation. You are free to write your own code in the main method to invoke the business methods to check its correctness.**

**Note:**

* In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
* Ensure to follow the object-oriented specifications provided in the question
* Ensure to provide the names for classes, attributes, and methods as specified in the question.
* Adhere to the code template, if provided.

**Sample Input/Output -1**

Enter the details

**Genoa:247:SpecialBeam:10**

LightHouse Location: Genoa

Height: 247

Light Type: SpecialBeam

Hour: 10

Flashes: 500

**Sample Input/Output -2**

Enter the details

**Vasto:307:StrongIntensity:-15**

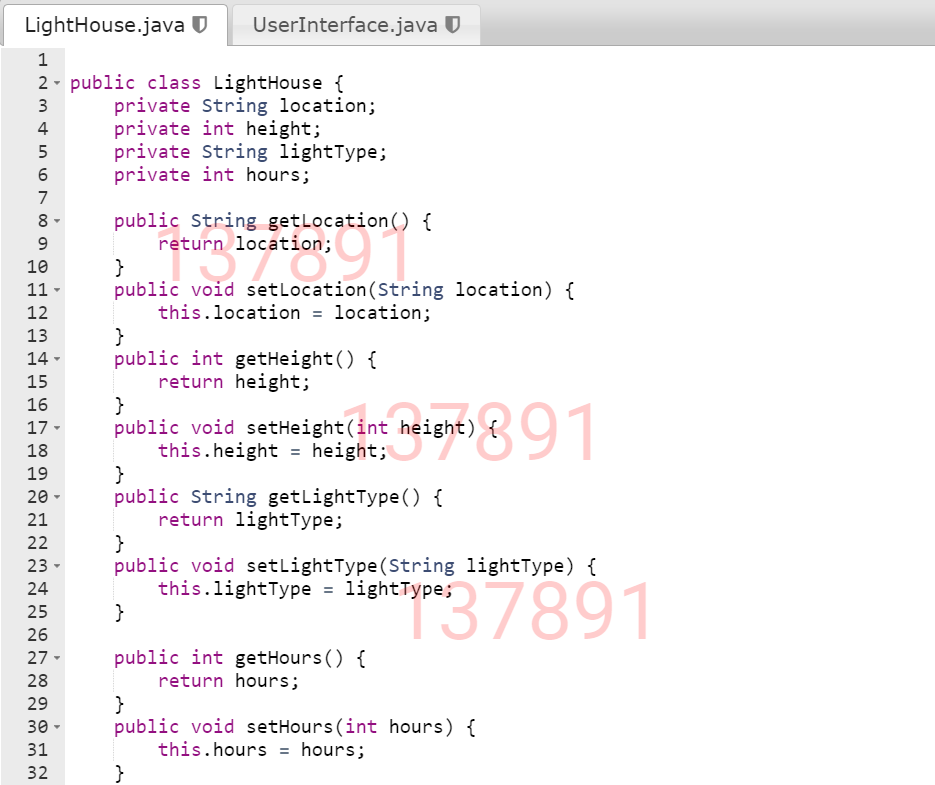
Invalid light house details

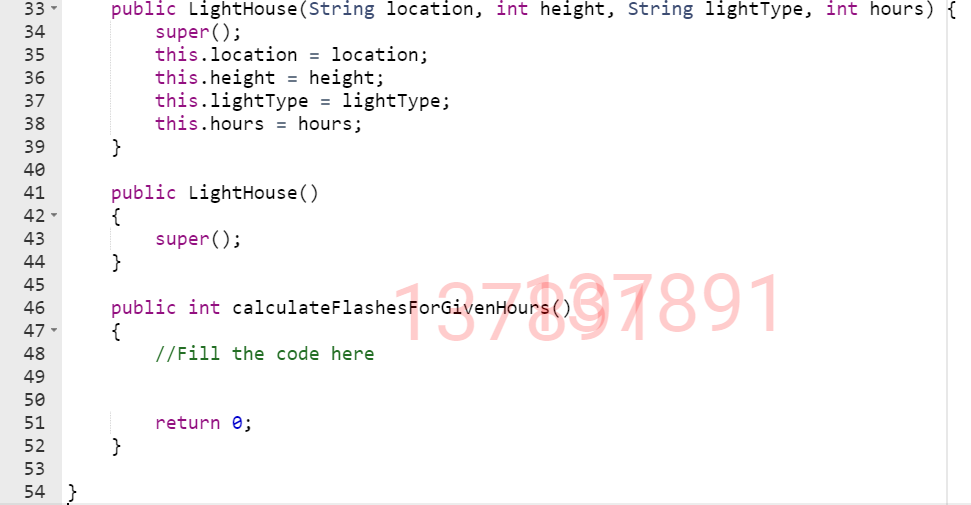
**Sample Input/Output -3**

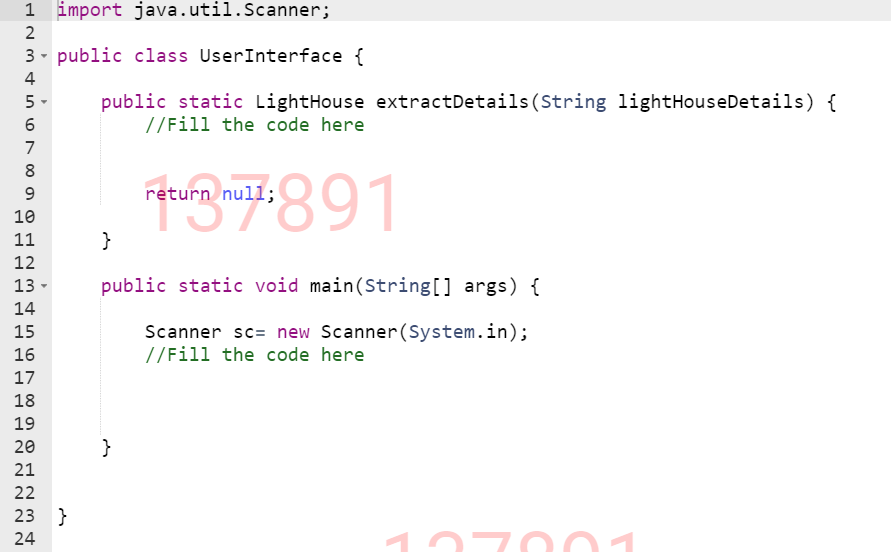
Enter the details

**Paris:-89:StrongIntensity:15**

Invalid light house details







**Beyond The Basics**

Beyond The Basics is one of the well-known Insurance companies, that wanted to automate the process of calculating the maturity amount of each policy, which will be easily accessed by their policyholders through their websites. As a developer help the company to implement the requirements.

**Component Specification: Insurance**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** |
| **Insurance** | String policyNumber  String insuranceProvider  double coverageAmount  String coverageType  int durationInYears | Necessary getters, setters and five argument constructors are provided as part of the code skeleton. |

**Functional Requirement 1: Extract the details of Insurance and create an object of the Insurance class.**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Method** | **Responsibilities** |
| **UserInterface** | public static Insurance **extractDetails**(String insuranceDetails) | This method should accept **insuranceDetails**as an argument and extracts the properties of **Insurance**from the argument and set these values to the **Insurance**object. |

**Functional Requirement 2: Calculate the maturity amount for the given duration.**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Method** | **Responsibilities** |
| **Insurance** | public double **calculateMaturityAmount**() | This method should calculate the maturity amount for the given duration in years and return the result.  If the durationInYears greater than zero then, calculate the maturityAmount.  ***Condition:***   * ***coverageAmount should be positive else return -1.*** * ***If durationsInYears Less than or equal to zero then, return -1.*** |

**Formula to calculate maturity amount**,

**MaturityAmount=coverageAmount**

**maturityAmount += (maturityAmount \* 0.05) ^durationInYear.**

**' ^ ' symbol represents power.**

Let, coverageAmount=1000, durationInYears=2

For 1st year,

MaturityAmount=1000+(1000\*0.05)=1050

For 2nd year,

MaturityAmount=1050+(1050\*0.05)=1102.50

**The main method in the UserInterface class is excluded from evaluation. You are free to write your own code in the main method to invoke the business methods to check its correctness.**

**Note:**

* In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
* Ensure to follow the object-oriented specifications provided in the question
* Ensure to provide the names for classes, attributes, and methods as specified in the question.
* Adhere to the code template, if provided.

**Sample Input 1**

Enter the details

**STR45789:Star Insurance:1000:Full:2**

**Sample Output 1**

Policy Number: STR45789

Insurance Provider: Star Insurance

Coverage Amount: 1000.0

Coverage Type: Full

Duration in years: 2

Maturity Amount: 1102.50

**Sample Input 2**

Enter the details

**ICIC102289:ICICI Insurance:10000:Half:-5**

**Sample Output 2**

Invalid details

**Explanation :**As durationInYears is invalid.

**Sample Input 2**

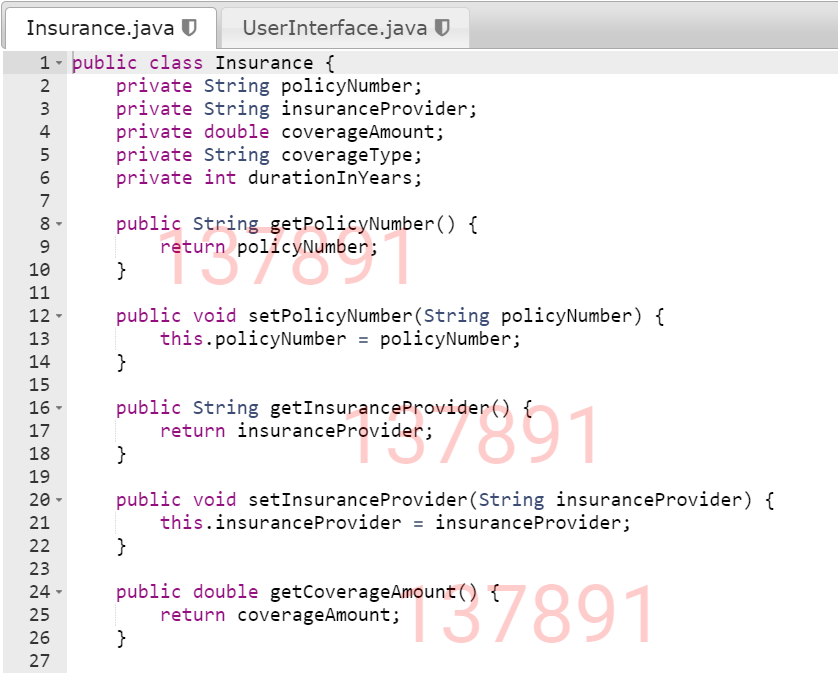
Enter the details

**ICIC132712:Prudential Insurance:-10000:Half:5**

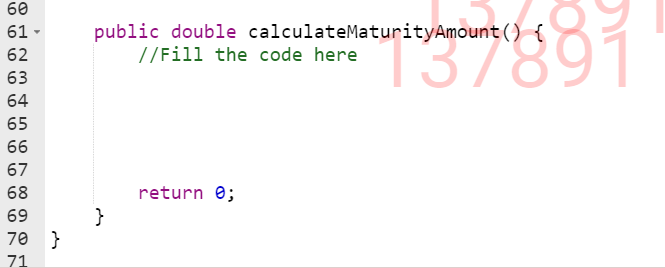
**Sample Output 2**

Invalid details

**Explanation :**As coverageAmount is invalid.







**The Phone Zone**

**"The Phone Zone"** is a popular mobile showroom in the city. They announced a massive New Year's Eve sale on 5G mobiles from five different mobile brands. Among them are **Oppo, Vivo, Samsung, OnePlus, and Realme**. They wanted to calculate the mobile price by deducting the discount percentage based on the phone brand. The manager of **"The Phone Zone"** seeks software developer assistance with their process. As a software developer, you create a Java program based on the requirements.

**Component Specification: MobileInfo**

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Attributes** | **Methods** |
| **MobileInfo** | String mobileName  String imeiNumber  String mobileBrand  double mobilePrice | Necessary getters, setters, and a four-argument constructor are provided as a part of the code skeleton. |

**Functional Requirement 1: Extract the details of the mobile and create an object of MobileInfo.**

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Methods** | **Responsibilities** |
| **UserInterface** | public static MobileInfo **extractDetails**(String mobileDetails) | This method accepts **mobileDetails**separated by colon as an argument and should extract the properties of the **MobileInfo**from the argument by parsing the **mobileDetails**. Set these values to the **MobileInfo** object and return this object . |

**Functional Requirement 2: Calculate the mobile price to be paid by the customer.**

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Methods** | **Responsibilities** |
| **MobileInfo** | public double **calculateMobilePriceToBePaid**() | This method is used to calculate the mobile price after deducting the discount%.  The IMEI number should contain exactly **15 digits** .  If the mobileBrand is "**Samsung**", the discount is 2% of the price.  If the mobileBrand is "**Realme**", the discount is 6% of the price.  If the mobileBrand is "**OnePlus**", the discount is 4% of the price.  If the mobileBrand is "**Oppo**", the discount is 5% of the price.  If the mobileBrand is "**Vivo**", the discount is 3% of the price.  ***Condition:***   * ***mobileBrand****is case-sensitive.* * *If the****mobileBrand****does not match any of the above types, return -1.* * *If the****mobilePrice****is less than or equal zero, return -1.* * *If the****imeiNumber****is less than or greater than 15 digits, return -1.* |

**Formula to calculate the mobile price to be paid:**  
Mobile price to be paid= mobilePrice -(( mobilePrice \*discount%)/100))

**Eg:** Let the mobileCost be 50000 ,  mobileBrand be Samsung. Discount for Samsung mobile is 2%,

Mobile price to be paid=50000-((50000\*2)/100) =50000-2000=48000.0

**The main method in the UserInterface class is excluded from the evaluation. You are free to write your own code in the main method to invoke the business methods to check its correctness.**

**Note:**

* In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
* Ensure to follow the object-oriented specifications provided in the question.
* Ensure to provide the names for classes, attributes, and methods as specified in the question.
* Adhere to the code template, if provided

**Sample Input/Output 1:**

Enter the Mobile details

**Reno7-5G:187654321114567:Oppo:35000**

Mobile Details

Mobile Name: Reno7-5G

Mobile IMEI Number: 187654321114567

Mobile Brand: Oppo

Mobile Price: 35000.0

Amount to be paid: 33250.0

**Sample Input/Output 2:**

Enter the Mobile details

**Moto M56:675434567825678:Motorola:28000**

Invalid Mobile details

**Sample Input/Output 3:**

Enter the Mobile details

**Real me 9 Pro:675434567823:Realme:29000**

Invalid Mobile details

**Sample Input/Output 4:**

Enter the Mobile details

**OnePlus Nord 5:675434567823890:OnePlus:-9**

Invalid Mobile details

