1. The festive season is about to begin. Mary is planning to purchase several items through an online shopping platform. She has added all the items to the shopping cart. Now she wants to verify if she has added all the items that she had planned to purchase. Help Mary to view the contents of the cart.

Implement the above scenario using an ArrayList.

Component Specification: Products

Type(Class)	**Attributes**	**Methods**	**Responsibilities**
Products	List<String> productList	Include the getter and setter method.	

Note: **The class and methods should be declared as public and all the attributes should be declared as private.**

Requirement 1: Add product to the ArrayList.

As per this requirement, the system should be able to add a product to the ArrayList.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Add a product to the ArrayList.	**Products**	**public void addProductToList(String product)**	This method is used to add a product to the ArrayList.

Requirement 2: Sort the products in the ArrayList in alphabetical order.

As per this requirement, the system should be able to sort the products in alphabetical order.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Sort the products in alphabetical order.	**Products**	**public void sortProductList()**	This method is used to sort the items in the ArrayList in alphabetical order.

In the **UserInterface** class,

1.Create a main method with the menu as described in the sample Input and Output.

2.When the user selects option **1.Add**, add the product into the productList.

3.When the user selects option **2.Display**, it should display the products in alphabetical order. If the list is empty, it should display **"The list is empty".**

4.When the user selects option **3.Exit**, display the message **"Thank you for using the application"** and terminate the program.

5.Option can take only values 1 to 3.

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.

- **Please don't use System.exit(0) to terminate the program.**

Sample Input / Output 1:

1.Add

2.Display

3.Exit

Enter your choice

1

Enter the product

purse

1.Add

2.Display

3.Exit

Enter your choice

1

Enter the product

key

1.Add

2.Display

3.Exit

Enter your choice

1

Enter the product

doll

1.Add

2.Display

3.Exit

Enter your choice

2

doll

key

purse

1.Add

2.Display

3.Exit

Enter your choice

3

Thank you for using the application

Sample Input / Output 2:

1.Add

2.Display

3.Exit

Enter your choice

2

The list is empty

1.Add

2.Display

3.Exit

Enter your choice

3

Thank you for using the application

Sample Input / Output 3:

1.Add

2.Display

3.Exit

Enter your choice

3

Thank you for using the application

2. Daniel is a Maths teacher. He often throws puzzles to his students to help them stay focused. One day he said, he will spell out a set of numbers. The students should exclude all the numbers that are divisible by five and six and then calculate the average of the remaining numbers and tell him. Help them to implement the above task.

Implement the above scenario using a TreeSet.

Component Specification: NumAvg

Type(Class)	**Attributes**	**Methods**	**Responsibilities**
NumAvg	TreeSet<Integer> numSet	Include the getter and setter method.	

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Requirement 1: Add number to the TreeSet.

As per this requirement, the system should be able to add a number to the TreeSet

Component Name	**Type (Class)**	**Methods**	**Responsibilities**
Add a number to the TreeSet	NumAvg	public void addNum(int num)	This method should add the number to the TreeSet only if it is not divisible by 5 and 6.

Requirement 2: Find the average of the numbers in the TreeSet.

As per this requirement, the system should be able to calculate the average of the numbers present in the TreeSet.

Component Specification: NumAvg

Component Name	**Type (Class)**	**Methods**	**Responsibilities**
Find the average of the numbers in the TreeSet.	NumAvg	public double calAvg()	This method is used to find the average of the numbers in the TreeSet.

In the UserInterface class,

1. Create a main method with the menu as described in the sample Input and Output.
2. When the user selects option **1.Add number**, add the number into numSet.
3. When the user selects option **2.Find Average**, it should display the average of the numbers of the numSet. If the set is empty, it should display **"The set is empty".**
4. When the user selects option **3.Exit**, display the message **"Thank you for using the application"** and terminate the program.
5. Option can take only values 1 to 3.

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.
- **Please don't use System.exit(0) to terminate the program.**

Sample Input / Output 1:

1.Add number

2.Find average

3.Exit

Enter your choice

1

Enter the number

7

1.Add number

2.Find average

3.Exit

Enter your choice

1

Enter the number

12

1.Add number

2.Find average

3.Exit

Enter your choice

1

Enter the number

13

1.Add number

2.Find average

3.Exit

Enter your choice

1

Enter the number

24

1.Add number

2.Find average

3.Exit

Enter your choice

2

10.0

1.Add number

2.Find average

3.Exit

Enter your choice

3

Thank you for using the application

Sample Input / Output 2:

1.Add number

2.Find average

3.Exit

Enter your choice

2

The set is empty

1.Add number

2.Find average

3.Exit

Enter your choice

3

Thank you for using the application

Sample Input / Output 3:

1.Add number

2.Find average

3.Exit

Enter your choice

3

Thank you for using the application

3. Remembering the names of the countries and their capitals can be very challenging. But children are generally fond of this particular challenge.

Help the children to implement the above task.

Implement the above scenario using a HashMap.

Component Specification: Country

Type(Class)	**Attributes**	**Methods**	**Responsibilities**
Country	Map<String,String> countryMap	Include the getter and setter method.	

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Requirement 1: Add a country and its capital to the HashMap.

As per this requirement, the system should be able to add a country and its capital to the HashMap.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Add a country and its capital to the HashMap.	Country	public void add(String cname, String capname)	This method should add a country and its capital to the HashMap. If a country already exists in the ***countryMap***, do not add it again.

Requirement 2: Search the capital of a given country.

As per this requirement, the system should be able to search the capital of a country from the HashMap.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Search the capital of a given country.	Country	public String search(String cname)	This method is used to search the capital of a country from the HashMap.

In the UserInterface class,

1. Create a main method with the menu as described in the sample Input and Output.
2. When the user selects option **1.Add**, add a country and its capital into HashMap countryMap.
3. When the user selects option **2.Search**, it should display the capital of a given country. If the map is empty, it should display **"The map is empty".** If the country is not available in the HashMap, it should display **"Data not found".**
4. When the user selects option **3.Exit**, display the message **"Thank you for using the application"** and terminate the program.
5. Option can take only values 1 to 3.

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.
- **Please don't use System.exit(0) to terminate the program.**

Sample Input / Output 1:

1.Add

2.Search

3.Exit

Enter your choice

1

Enter the country name

Italy

Enter the capital name

Rome

1.Add

2.Search

3.Exit

Enter your choice

1

Enter the country name

New Zealand

Enter the capital name

Wellington

1.Add

2.Search

3.Exit

Enter your choice

1

Enter the country name

Russia

Enter the capital name

Moscow

1.Add

2.Search

3.Exit

Enter your choice

2

Enter the country name

New Zealand

Wellington

1.Add

2.Search

3.Exit

Enter your choice

3

Thank you for using the application

Sample Input / Output 2:

1.Add

2.Search

3.Exit

Enter your choice

1

Enter the country name

Russia

Enter the capital name

Moscow

1.Add

2.Search

3.Exit

Enter your choice

1

Enter the country name

Italy

Enter the capital name

Rome

1.Add

2.Search

3.Exit

Enter your choice

2

Enter the country name

Singapore

Data not found

1.Add

2.Search

3.Exit

Enter your choice

3

Thank you for using the application

Sample Input / Output 3:

1.Add

2.Search

3.Exit

Enter your choice

2

The map is empty

1.Add

2.Search

3.Exit

Enter your choice

3

Thank you for using the application

Sample Input / Output 4:

1.Add

2.Search

3.Exit

Enter your choice

3

Thank you for using the application

4. Various competitions are conducted in schools by grouping the students into different houses or teams.

Write a java program to implement the above scenario.

The above scenario should be implemented using a HashSet.

Component Specification: ClassHouseFormation

Type(Class)	**Attributes**	**Methods**	**Responsibilities**
ClassHouseFormation	HashSet<String> studSet	Include the getter and setter method.	

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Requirement 1: Add name to the HashSet.

As per this requirement, the system should be able to add a student name to the HashSet.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Add a student name to the HashSet.	ClassHouseFormation	public void addName(String details)	This method should extract the student name from the given input and add only the name to the HashSet. **Eg input**: 101:Alan The name alone should be extracted and added to the HashSet. **(names are case sensitive)**

Requirement 2: Find the house of student names in the HashSet.

As per this requirement, the system should be able to find the house of the student names present in the HashSet and append the house name to respective stiudent name.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Find the house of the names in the HashSet	ClassHouseFormation	public HashSet<String> formTeam()	This method is used to find the house of the student names in the HashSet. Names that start with letters A - H belong to house "RED". Names that start with letters I - P belong to house "BLUE". Names that start with letters Q - Z belong to house "GREEN". **(House name is case sensitive)** Append the student name and the house name. **Eg:** If name is Alan and house is RED, the appended string should be Alan:RED. Store the appended string in another HashSet and return the same.

In the UserInterface class, call the above methods and display the output as given in the sample input/output.

Note:

- The number of students should be greater than 0. Display "**Invalid input**" if this condition fails.
- 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.
- **Please don't use System.exit(0) to terminate the program.**

Sample Input / Output 1:

Enter the number of students

4

Enter the details

101:Daniel

102:John

103:Michael

104:Steve

Daniel:RED

John:BLUE

Steve:GREEN

Michael:BLUE

Sample Input / Output 2:

Enter the number of students

3

Enter the details

101:Alan

102:Diana

103:Elizabeth

Alan:RED

Diana:RED

Elizabeth:RED

Sample Input / Output 3:

Enter the number of students

-1

Invalid input

5. Cars have become a necessity of life. People's interest in buying cars is growing like never before.

The government is planning to take a survey on cars for the previous year. The survey needs to have various options with which car details can be fetched.

Help the government to implement the above task.

Implement the above scenario using a TreeMap.

Component Specification: Car

Type(Class)	**Attributes**	**Methods**	**Responsibilities**
Car	Map<String,Integer> carMap	Include the getter and setter method.	

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Requirement 1: Add a car name and number of cars sold into the TreeMap.

As per this requirement, the system should be able to add a car name and number of cars sold into the TreeMap.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Add a car name and number of cars sold into the TreeMap.	Car	public void addCar(String name,int num)	This method should add a car name and number of cars sold into the TreeMap. If a car name already exists in the ***carMap***, do not add it again.

Requirement 2: Display the number of cars sold when a car name is given.

As per this requirement, the system should be able to search for a car from the TreeMap and return the number of cars sold.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Search a car from the TreeMap.	Car	public int carByName(String name)	This method is used to search for a car from the TreeMap and return the number of cars sold. Else return -1.

Requirement 3: Display the car names when a count of cars sold is given.

As per this requirement, the system should be able to search for cars from the TreeMap.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Search for cars from the TreeMap.	Car	public List<String> carByCount(int count)	This method is used to search for cars sold from the TreeMap that is greater than or equal to the given count and return the same in a list.

Requirement 4: Display the total number of cars sold.

As per this requirement, the system should be able to find the total number of cars sold from the TreeMap.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Calculate the total number of cars sold.	Car	public int totalCarsSold()	This method is used to calculate the total number of cars sold from the TreeMap and return the same.

In the UserInterface class,

1. Create a main method with the menu as described in the sample Input and Output.
2. When the user selects option **1.Add car**, add a car name and the number of cars sold into the carMap.
3. When the user selects option **2.Search by name**, it should display the count of cars sold for a given car name. If the map is empty, it should display **"The map is empty".** If the car is not available in the Map, it should display **"Car not found".**
4. When the user selects option **3.Search by count**, it should display the names of cars sold that are greater than or equal to a given count. If the map is empty, it should display **"The map is empty".** If no such cars are available in the Map, it should display **"No cars found".**
5. When the user selects option **4.Total**, it should display the total number of cars sold. If the map is empty, it should display **"The map is empty".**
6. When the user selects option **5.Exit**, display the message **"Thank you for using the application"** and terminate the program.
7. Option can take only values 1 to 5.

Note:

- **Car name is case insensitive.**
- **The value to be given for number of cars sold should be a positive number.**
- 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.
- **Please don't use System.exit(0) to terminate the program.**

Sample Input / Output 1:

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

1

Enter the car name

Alto

Enter no of cars sold

2500

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

1

Enter the car name

Wagon R

Enter no of cars sold

2000

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

1

Enter the car name

Duster

Enter no of cars sold

700

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

2

Enter the car name

Duster

700

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

3

Enter the count

2300

Alto

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

4

5200

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

5

Thank you for using the application

Sample Input / Output 2:

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

1

Enter the car name

Jazz

Enter no of cars sold

350

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

2

Enter the car name

Beat

Car not found

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

3

Enter the count

500

No cars found

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

5

Thank you for using the application

Sample Input / Output 3:

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

2

The map is empty

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

3

The map is empty

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

4

The map is empty

1.Add car

2.Search by name

3.Search by count

4.Total

5.Exit

Enter your choice

5

Thank you for using the application

. Fresh Farm is one of the best producers of fresh farm products in the city. Their products include dairy products, vegetables, and fruits. They want an estimate of the maximum product which is sold from the farm each day. Develop a java application for the same using Streams.

Component Specification: Carton (POJO Class)

Type (Class)	Attributes	Methods
Carton	String productName int quantity double productCost	Getters and Setters are given in the code skeleton.

Component Specification: CartonUtility

Type (Class)	Attributes	Methods	Responsibilities
CartonUtility	List<Carton> cartonList		Provide the setter and getter for the class attribute.
CartonUtility		public Stream<Carton> convertToStream()	Convert the list of Carton objects to the stream of Carton and return it.
CartonUtility		public Carton findMax(Stream<Carton> stream1)	Find the maximum quantity of product sold for that day and return that object.

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Note:

- The number of cartons should be a valid natural number, if fails print as "Invalid".
- The number of quantities should be a valid natural number, if fails print as "Quantity number should be a valid number".
- Input records are entered in the format: productName/ quantity/ productCost.
- 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 description.

- Ensure to provide the names for classes, attributes, and methods as specified in the question description.
- Adhere to the code template, if provided.

Sample Input/Output 1:

Enter the number of cartons

3

Enter carton details

egg/40/300

brinjal/50/600

turnip/60/900

turnip had the highest quantity with 60 nos

Sample Input/Output 2:

Enter the number of cartons

2

Enter carton details

brinjal/0/0

Quantity number should be a valid number

Sample Input/Output 3:

Enter the number of cartons

0

Invalid

7.

A famous fruit stall in the marketplace approaches you to create an application in which their customers can estimate the total bill amount for the fruits in the basket. As a Java

developer, create a Java application to add fruits to the basket and calculate the bill amount.

Component Specification: FruitBasket (POJO Class)

Type (Class)	Attributes	Methods
FruitBasket	String fruitName int weightInKgs int pricePerKg	Getters and Setters as well as three arguments constructor are given in the code skeleton.

Component Specification: FruitBasketUtility

Type (Class)	Methods	Responsibilities
FruitBasketUtility	List<FruitBasket> fruitBasketList	Getters and Setters are given in the code skeleton.
FruitBasketUtility	public void addToBasket(FruitBasket fbObj)	This method of the FruitBasketUtility class adds the FruitBasket object into the fruitBasketList.
FruitBasketUtility	public int calculateBill(Stream<FruitBasket> fruitBasketStream)	This method calculates the total bill amount from the Stream of FruitBasket objects. Each object will have a fruit detail. On multiplying the weightInKgs and pricePerKg of each FruitBasket object, the individual bill amount for that particular fruit can be calculated. This method should return the total bill amount by adding each individual bill amount of fruits present in the Stream.

Note: The class and methods should be declared as public.

In the UserInterface class for retrieving the list of FruitBasket objects from the FruitBasketUtility class and converting the List of objects into a Stream of objects. Then pass the Stream of FruitBasket objects to the calculateBill method in the FruitBasketUtility class and display the total bill amount.

User Interface:

- Display the options to choose as "Select an option: 1. Add Fruit to Basket 2. Calculate Bill 3. Exit" for all iterations.
- Any valid option can be chosen as 1 or 2 or 3. Otherwise, display a message as "Invalid option. Please try again." and continue to display the options.

- For option 1: get the fruitName, weightInKgs and pricePerKg as inputs to process the functional requirements and continue to display the options.
- For option 2: retrieve the list as mentioned in the functional requirements. If the retrieved list is empty, then display "Your basket is empty. Please add fruits." and continue to display the options. Otherwise, display the total bill amount as "The estimated bill amount is Rs <total bill amount>" and continue to display the options.
- For option 3: display a message "Thank you for using the application" and terminate the program.

Note:

- In the Sample Inputs/ Outputs provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
- Few of the User Interface requirements will be provided in the code template itself. Adhere to the code template. Enclose your code in the respective required blocks alone.
- Do not edit or delete the codes provided in the code template.
- Adhere to the Sample Inputs/ Outputs.

Do not use System.exit(0) for terminating the program.

Sample Input/ Output 1:

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

2

Your basket is empty. Please add fruits.

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

1

Enter the fruit name

Mango

Enter weight in Kgs

5

Enter price per Kg

30

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

2

The estimated bill amount is Rs 150

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

1

Enter the fruit name

Apple

Enter weight in Kgs

2

Enter price per Kg

80

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

1

Enter the fruit name

Kiwi

Enter weight in Kgs

3

Enter price per Kg

45

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

2

The estimated bill amount is Rs 445

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

3

Thank you for using the application.

Sample Input/ Output 2:

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

4

Invalid option. Please try again.

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

2

Your basket is empty. Please add fruits.

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

3

Thank you for using the application.