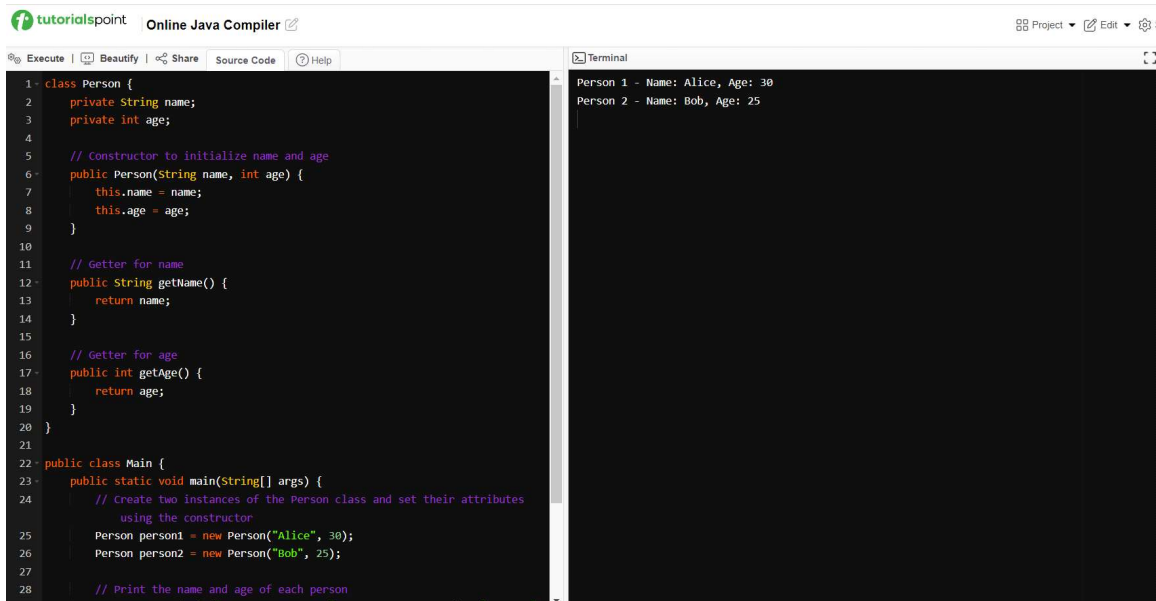


1. Write a Java program to create a class called "Person" with a name and age attribute. Create two instances of the "Person" class, set their attributes using the constructor, and print their name and age.



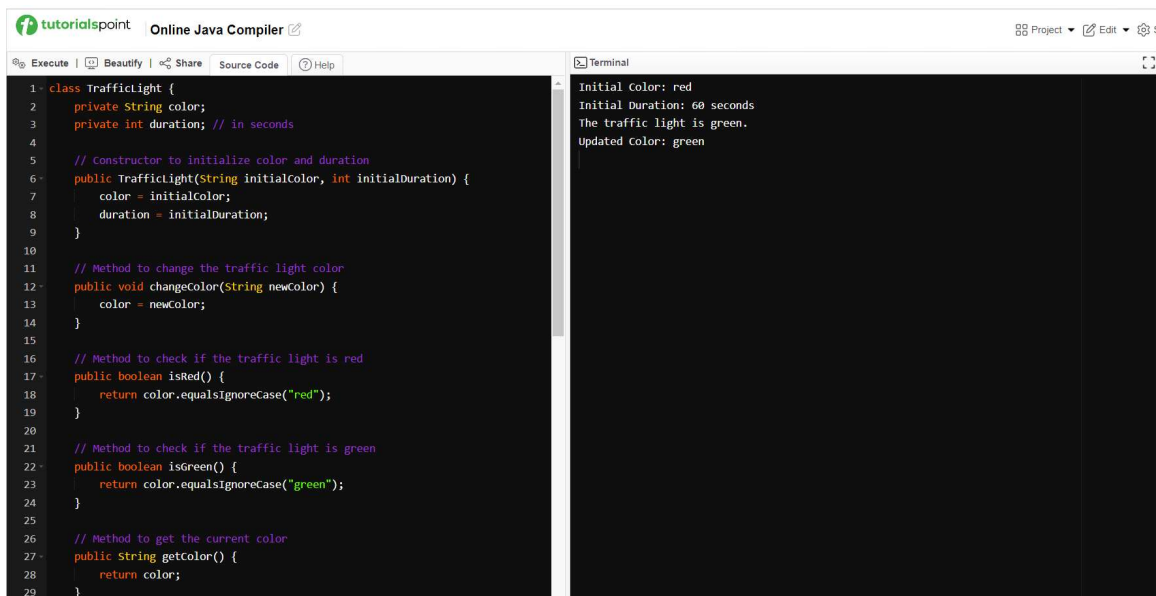
The screenshot shows the 'Online Java Compiler' interface. The code editor on the left contains the following Java code:

```
1- class Person {
2-     private String name;
3-     private int age;
4-
5-     // Constructor to initialize name and age
6-     public Person(String name, int age) {
7-         this.name = name;
8-         this.age = age;
9-     }
10-
11-     // Getter for name
12-     public String getName() {
13-         return name;
14-     }
15-
16-     // Getter for age
17-     public int getAge() {
18-         return age;
19-     }
20- }
21-
22- public class Main {
23-     public static void main(String[] args) {
24-         // Create two instances of the Person class and set their attributes
25-         // using the constructor
26-         Person person1 = new Person("Alice", 30);
27-         Person person2 = new Person("Bob", 25);
28-
29-         // Print the name and age of each person
```

The terminal window on the right shows the output of the program:

```
Person 1 - Name: Alice, Age: 30
Person 2 - Name: Bob, Age: 25
```

2. Write a Java program to create class called "TrafficLight" with attributes for color and duration, and methods to change the color and check for red or green



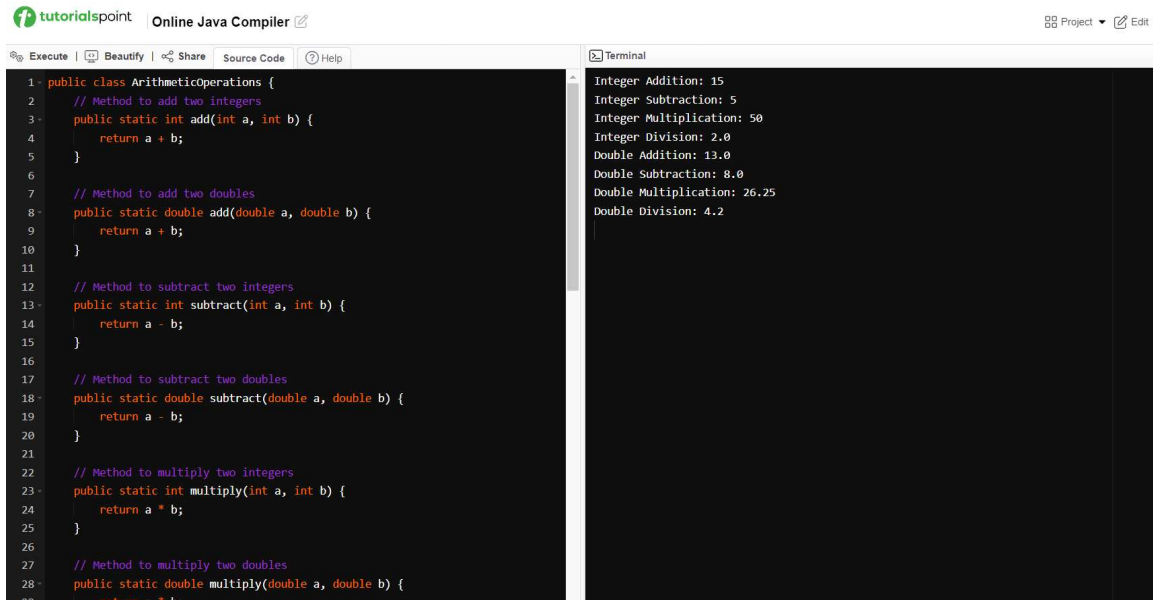
The screenshot shows the 'Online Java Compiler' interface. The code editor on the left contains the following Java code:

```
1- class TrafficLight {
2-     private String color;
3-     private int duration; // in seconds
4-
5-     // constructor to initialize color and duration
6-     public TrafficLight(String initialColor, int initialDuration) {
7-         color = initialColor;
8-         duration = initialDuration;
9-     }
10-
11-     // Method to change the traffic light color
12-     public void changeColor(String newColor) {
13-         color = newColor;
14-     }
15-
16-     // Method to check if the traffic light is red
17-     public boolean isRed() {
18-         return color.equalsIgnoreCase("red");
19-     }
20-
21-     // Method to check if the traffic light is green
22-     public boolean isGreen() {
23-         return color.equalsIgnoreCase("green");
24-     }
25-
26-     // Method to get the current color
27-     public String getColor() {
28-         return color;
29-     }
30- }
```

The terminal window on the right shows the output of the program:

```
Initial Color: red
Initial Duration: 60 seconds
The traffic light is green.
Updated color: green
```

3. Write a Java program to perform arithmetic operations using method overloading.

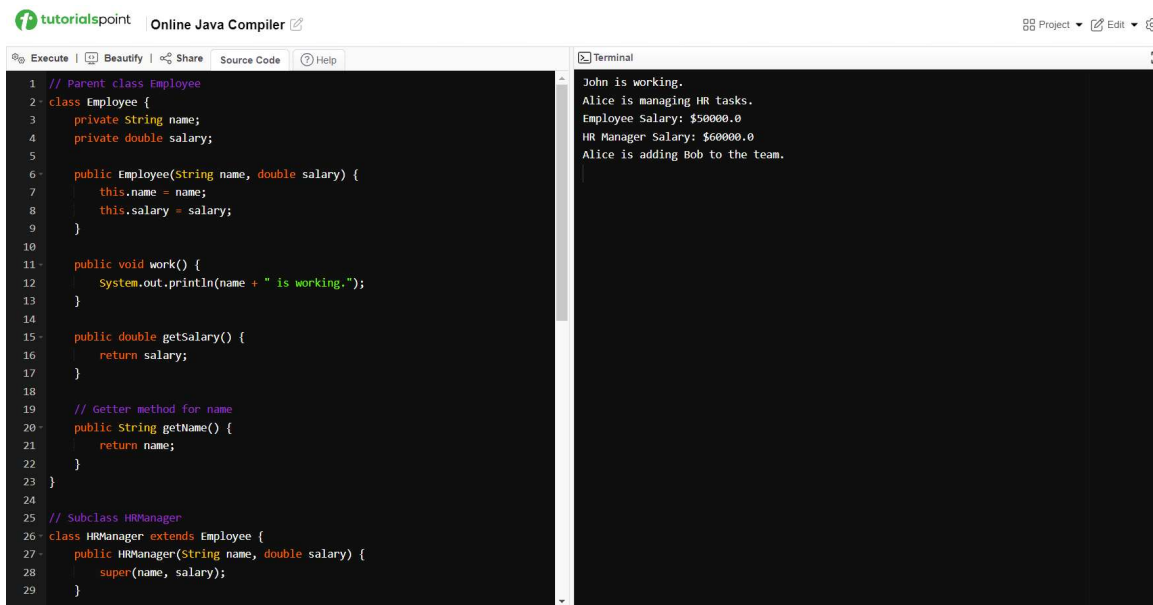


The screenshot shows the 'Online Java Compiler' interface. The source code on the left defines a class `ArithmeticOperations` with static methods for addition and subtraction of integers and doubles. The terminal on the right displays the output of these operations.

```
1- public class ArithmeticOperations {
2-     // Method to add two integers
3-     public static int add(int a, int b) {
4-         return a + b;
5-     }
6-
7-     // Method to add two doubles
8-     public static double add(double a, double b) {
9-         return a + b;
10-    }
11-
12-    // Method to subtract two integers
13-    public static int subtract(int a, int b) {
14-        return a - b;
15-    }
16-
17-    // Method to subtract two doubles
18-    public static double subtract(double a, double b) {
19-        return a - b;
20-    }
21-
22-    // Method to multiply two integers
23-    public static int multiply(int a, int b) {
24-        return a * b;
25-    }
26-
27-    // Method to multiply two doubles
28-    public static double multiply(double a, double b) {
29-        return a * b;
30-    }
31-}
```

```
Integer Addition: 15
Integer Subtraction: 5
Integer Multiplication: 50
Integer Division: 2.0
Double Addition: 13.0
Double Subtraction: 8.0
Double Multiplication: 26.25
Double Division: 4.2
```

4) Write a Java program to create a class called `Employee` with methods called `work()` and `getSalary()`. Create a subclass called `HRManager` that overrides the `work()` method and adds a new method called `addEmployee()`.



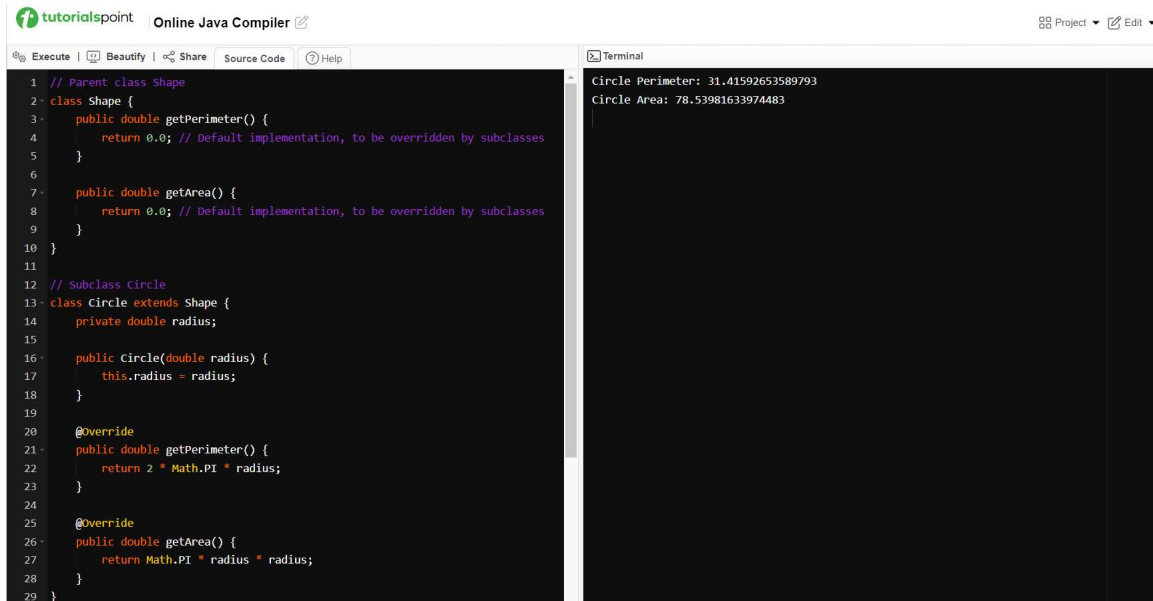
The screenshot shows the 'Online Java Compiler' interface. The source code on the left defines a base class `Employee` and a subclass `HRManager` that extends it. The terminal on the right displays the output of the program.

```
1 // Parent class Employee
2 class Employee {
3     private String name;
4     private double salary;
5
6     public Employee(String name, double salary) {
7         this.name = name;
8         this.salary = salary;
9     }
10
11     public void work() {
12         System.out.println(name + " is working.");
13     }
14
15     public double getSalary() {
16         return salary;
17     }
18
19     // Getter method for name
20     public String getName() {
21         return name;
22     }
23 }
24
25 // Subclass HRManager
26 class HRManager extends Employee {
27     public HRManager(String name, double salary) {
28         super(name, salary);
29     }
30 }
```

```
John is working.
Alice is managing HR tasks.
Employee Salary: $50000.0
HR Manager Salary: $60000.0
Alice is adding Bob to the team.
```

5) Write a Java program to create a class called `Shape` with methods called `getPerimeter()` and

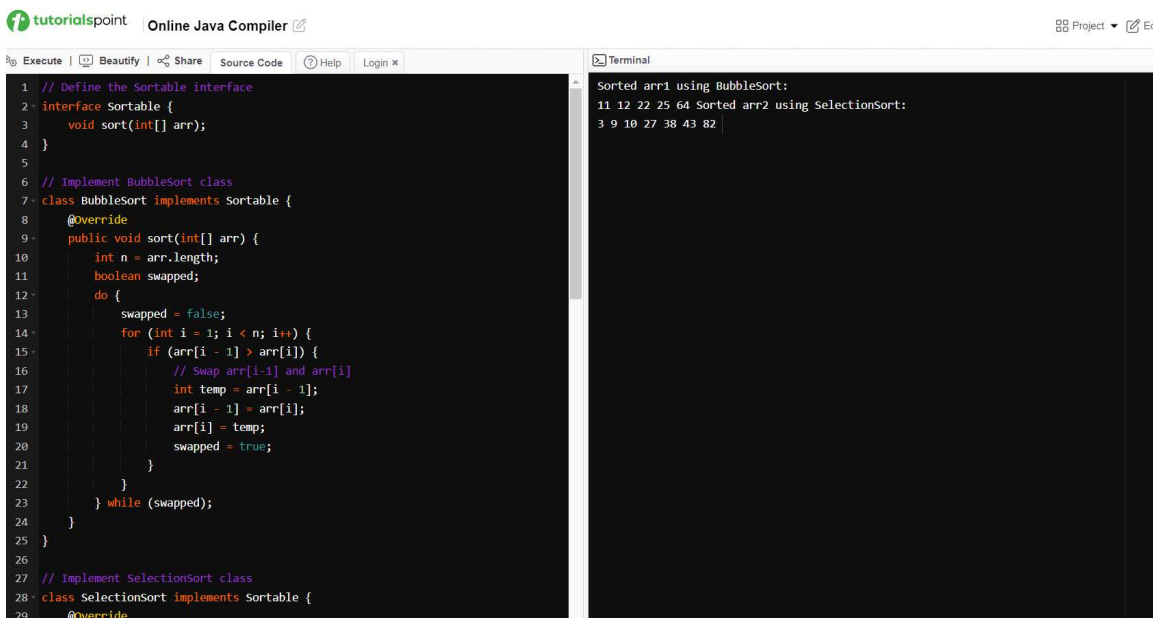
getArea()). Create a subclass called Circle that overrides the getPerimeter() and getArea() methods to calculate the area and perimeter of a circle.



```
1 // Parent class Shape
2 class Shape {
3     public double getPerimeter() {
4         return 0.0; // Default implementation, to be overridden by subclasses
5     }
6
7     public double getArea() {
8         return 0.0; // Default implementation, to be overridden by subclasses
9     }
10 }
11
12 // Subclass Circle
13 class Circle extends Shape {
14     private double radius;
15
16     public Circle(double radius) {
17         this.radius = radius;
18     }
19
20     @Override
21     public double getPerimeter() {
22         return 2 * Math.PI * radius;
23     }
24
25     @Override
26     public double getArea() {
27         return Math.PI * radius * radius;
28     }
29 }
```

Circle Perimeter: 31.41592653589793  
Circle Area: 78.53981633974483

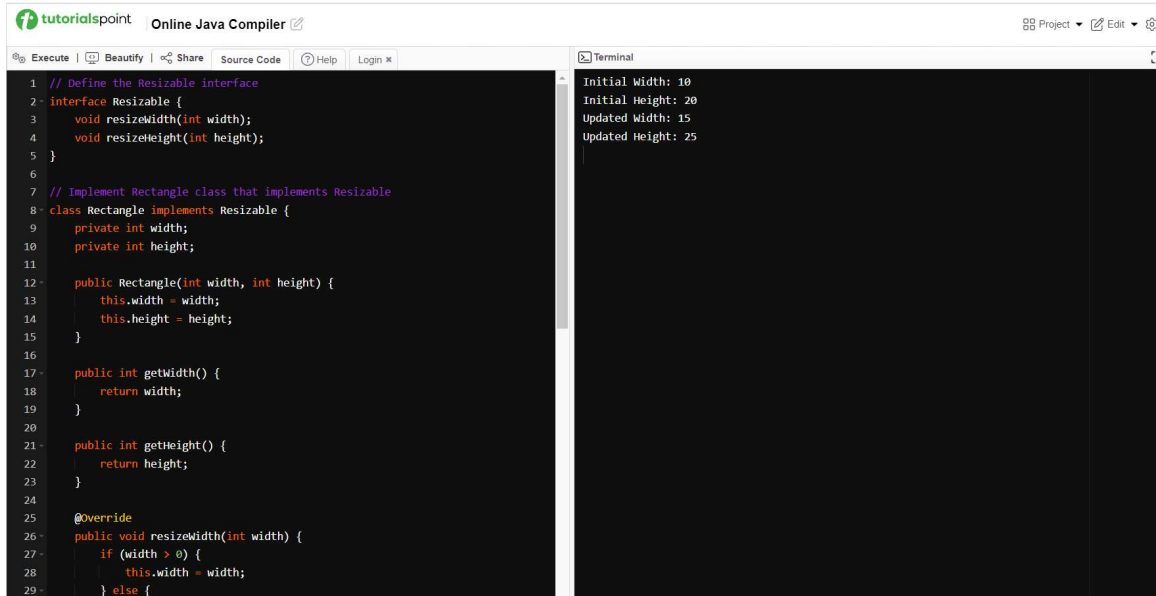
6. Write a Java program to create an interface Sortable with a method sort() that sorts an array of integers in ascending order. Create two classes BubbleSort and SelectionSort that implement the Sortable interface and provide their own implementations of the sort() method.



```
1 // Define the Sortable interface
2 interface Sortable {
3     void sort(int[] arr);
4 }
5
6 // Implement BubbleSort class
7 class BubbleSort implements Sortable {
8     @Override
9     public void sort(int[] arr) {
10         int n = arr.length;
11         boolean swapped;
12         do {
13             swapped = false;
14             for (int i = 1; i < n; i++) {
15                 if (arr[i - 1] > arr[i]) {
16                     // Swap arr[i-1] and arr[i]
17                     int temp = arr[i - 1];
18                     arr[i - 1] = arr[i];
19                     arr[i] = temp;
20                     swapped = true;
21                 }
22             }
23         } while (swapped);
24     }
25 }
26
27 // Implement SelectionSort class
28 class SelectionSort implements Sortable {
29     @Override
```

Sorted arr1 using BubbleSort:  
11 12 22 25 64 Sorted arr2 using SelectionSort:  
3 9 10 27 38 43 82

7. Write a Java program to create an interface Resizable with methods `resizeWidth(int width)` and `resizeHeight(int height)` that allow an object to be resized. Create a class `Rectangle` that implements the `Resizable` interface and implements the resize methods.



```
1 // Define the Resizable interface
2 interface Resizable {
3     void resizeWidth(int width);
4     void resizeHeight(int height);
5 }
6
7 // Implement Rectangle class that implements Resizable
8 class Rectangle implements Resizable {
9     private int width;
10    private int height;
11
12    public Rectangle(int width, int height) {
13        this.width = width;
14        this.height = height;
15    }
16
17    public int getWidth() {
18        return width;
19    }
20
21    public int getHeight() {
22        return height;
23    }
24
25    @Override
26    public void resizeWidth(int width) {
27        if (width > 0) {
28            this.width = width;
29        } else {
```

```
Initial Width: 10
Initial Height: 20
Updated Width: 15
Updated Height: 25
```

8. Write a Java program to create an interface `Flyable` with a method called `fly_obj()`. Create three classes `Spacecraft`, `Airplane`, and `Helicopter` that implement the `Flyable` interface. Implement the `fly_obj()` method for each of the three classes. Hint :- `fly_obj` definition – prints the particular object is flying

```

1 // Define the Flyable interface
2 interface Flyable {
3     void fly_obj();
4 }
5
6 // Implement Spacecraft class
7 class Spacecraft implements Flyable {
8     @Override
9     public void fly_obj() {
10         System.out.println("Spacecraft is flying in outer space.");
11     }
12 }
13
14 // Implement Airplane class
15 class Airplane implements Flyable {
16     @Override
17     public void fly_obj() {
18         System.out.println("Airplane is flying in the sky.");
19     }
20 }
21
22 // Implement Helicopter class
23 class Helicopter implements Flyable {
24     @Override
25     public void fly_obj() {
26         System.out.println("Helicopter is flying in the air.");
27     }
28 }
29
30 public class Main {

```

Spacecraft is flying in outer space.  
Airplane is flying in the sky.  
Helicopter is flying in the air.

9. Write a Java program to have the arithmetic functions defined in different user-defined packages and incorporate all the packages and perform the function in a single class.

```

10 package com.myapp.arithmetic;
11
12 public class Addition {
13     public static int add(int a, int b) {
14         return a + b;
15     }
16 }
17
18 package com.myapp.arithmetic;
19
20 public class Subtraction {
21     public static int subtract(int a, int b) {
22         return a - b;
23     }
24 }
25
26 package com.myapp.arithmetic;
27
28 public class Multiplication {
29     public static int multiply(int a, int b) {
30         return a * b;
31     }
32 }
33 // com.myapp.Main.java
34 package com.myapp;
35
36 import com.myapp.arithmetic.Addition;
37 import com.myapp.arithmetic.Subtraction;
38 import com.myapp.arithmetic.Multiplication;

```

10. Create two different packages to compute bubblesort and selection sort. Write a Java program to implement sorting functions in a single class.

tutorialspointOnline Java Compiler

ProjectEdit

Execute | Beautify | Share | Source Code | Help

Terminal

```
10 // BubbleSort.java in com.myapp.bubblesort package
11 package com.myapp.bubblesort;
12
13 public class BubbleSort {
14     public static void sort(int[] arr) {
15         int n = arr.length;
16         boolean swapped;
17         do {
18             swapped = false;
19             for (int i = 1; i < n; i++) {
20                 if (arr[i - 1] > arr[i]) {
21
22                     int temp = arr[i - 1];
23                     arr[i - 1] = arr[i];
24                     arr[i] = temp;
25                     swapped = true;
26                 }
27             }
28         } while (swapped);
29     }
30 }
31
32 package com.myapp;
33
34 import com.myapp.bubblesort.BubbleSort;
35 import com.myapp.selectionsort.SelectionSort;
36 import java.util.Arrays;
37
38 public class Main {
39     public static void main(String[] args) {
```