`printf`

(//)

/*

*/

/**

```
public class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Prints the string to
    the console.
    }
}
```

.java

HelloWorldApp.java

.class

HelloWorldApp.class

public

private

protected

SecurityException

static

void

```
System.exit()
```

String

args

```
public static void main(String... args)
```

```
public static void main(String[])
```

```
String[] args
```

main

System

out

PrintStream

```
println(String)
```

```java
// This is an example of a single line comment using two slashes
\par
/*
 * This is an example of a multiple line comment using the slash and
     asterisk.
 * This type of comment can be used to hold a lot of information or
    deactivate
 * code, but it is very important to remember to close the comment.
 */
\par
package fibsandlies;
\par
import java.util.Map;
import java.util.HashMap;
\par
/**
 * This is an example of a Javadoc comment; Javadoc can compile
    documentation
 * from this text. Javadoc comments must immediately precede the
    class, method,
 * or field being documented.
 * @author Wikipedia Volunteers
 */
public class FibCalculator extends Fibonacci implements Calculator {
    private static Map<Integer, Integer> memoized = new HashMap<>();
\par
/*
     * The main method written as follows is used by the JVM as a
    starting point
     * for the program.
     */
    public static void main(String[] args) {
        memoized.put(1, 1);
        memoized.put(2, 1);
        System.out.println(fibonacci(12)); // Get the 12th Fibonacci
     number and print to console
    }
\par
/**
     * An example of a method written in Java, wrapped in a class.
     * Given a non-negative number FIBINDEX, returns
     * the Nth Fibonacci number, where N equals FIBINDEX.
     *
     * @param fibIndex The index of the Fibonacci number
     * @return the Fibonacci number
     */
    public static int fibonacci(int fibIndex) {
        if (memoized.containsKey(fibIndex)) return memoized.get(
    fibIndex);
        else {
            int answer = fibonacci(fibIndex - 1) + fibonacci(
    fibIndex - 2);
            memoized.put(fibIndex, answer);
            return answer;
        }
    }
}
\par
```

service()

HttpServlet

GenericServlet

doGet(), doPost(), doPut(), doDelete()

```java
import java.io.IOException;
\par
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
\par
public class ServletLifeCycleExample extends HttpServlet {
\par
private Integer sharedCounter;
\par
@Override
    public void init(final ServletConfig config) throws
    ServletException {
        super.init(config);
        getServletContext().log("init() called");
        sharedCounter = 0;
    }
\par
@Override
    protected void service(final HttpServletRequest request, final
    HttpServletResponse response) throws ServletException,
    IOException {
        getServletContext().log("service() called");
\par
int localCounter;
\par
synchronized (sharedCounter) {
              sharedCounter++;
\par
localCounter = sharedCounter;
        }
\par
response.getWriter().write("Incrementing the count to " +
    localCounter);  // accessing a local variable
    }
\par
@Override
    public void destroy() {
        getServletContext().log("destroy() called");
    }
}
```

```
<p>Counting to three:</p>
<% for (int i=1; i<4; i++) { %>
    <p>This number is <%= i %>.</p>
<% } %>
<p>OK.</p>
```

```
Counting to three:
\par
This number is 1.
\par
This number is 2.
\par
This number is 3.
\par
OK.
```

```
// Hello.java (Java SE 5)
import javax.swing.*;
\par
public class Hello extends JFrame {
    public Hello() {
        super("hello");
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE)
    ;
        this.add(new JLabel("Hello, world!"));
        this.pack();
        this.setVisible(true);
    }
\par
public static void main(final String[] args) {
        new Hello();
    }
}
```

`import`

javax.swing

Hello

extends

JFrame

Hello()

```
setDefaultCloseOperation(int)
```

```
WindowConstants.EXIT_ON_CLOSE
```

JLabel

```
add(Component)
```

Container

pack()

Window

```
main()
```

```
setVisible(boolean)
```

Component

true

```
import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.WindowConstants;
import javax.swing.SwingUtilities;
\par
public class SwingExample implements Runnable {
\par
@Override
    public void run() {
        // Create the window
        JFrame f = new JFrame("Hello, !");
        // Sets the behavior for when the window is closed
        f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        // Add a layout manager so that the button is not placed on
    top of the label
        f.setLayout(new FlowLayout());
        // Add a label and a button
        f.add(new JLabel("Hello, world!"));
        f.add(new JButton("Press me!"));
        // Arrange the components inside the window
        f.pack();
        // By default, the window is not visible. Make it visible.
        f.setVisible(true);
    }
\par
public static void main(String[] args) {
        SwingExample se = new SwingExample();
        // Schedules the application to be run at the correct time
    in the event queue.
        SwingUtilities.invokeLater(se);
    }
\par
}
```

SwingUtilities.invokeLater(Runnable))

```
package javafxtuts;
\par
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
\par
public class JavaFxTuts extends Application {
    public JavaFxTuts() {
        //Optional constructor
    }
    @Override
    public void init() {
         //By default this does nothing, but it
         //can carry out code to set up your app.
         //It runs once before the start method,
         //and after the constructor.
    }
\par
@Override
    public void start(Stage primaryStage) {
        // Creating the Java button
        final Button button = new Button();
        // Setting text to button
        button.setText("Hello World");
        // Registering a handler for button
        button.setOnAction((ActionEvent event) -> {
            // Printing Hello World! to the console
            System.out.println("Hello World!");
        });
        // Initializing the StackPane class
        final StackPane root = new StackPane();
        // Adding all the nodes to the StackPane
        root.getChildren().add(button);
        // Creating a scene object
        final Scene scene = new Scene(root, 300, 250);
        // Adding the title to the window (primaryStage)
        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        // Show the window(primaryStage)
        primaryStage.show();
    }
    @Override
    public void stop() {
        //By default this does nothing
        //It runs if the user clicks the go-away button
        //closing the window or if Platform.exit() is called.
        //Use Platform.exit() instead of System.exit(0).
        //This is where you should offer to save any unsaved
        //stuff that the user may have generated.
    }
\par
/**
     * Main function that opens the "Hello World!" window
     *
     * @param arguments the command line arguments
     */
    public static void main(final String[] arguments) {
        launch(arguments);
    }
}
```

```
// Your First Program
\par
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

```
javac HelloWorld.java
java HelloWorld
```

```
java HelloWorld
```

```
class HelloWorld {
... .. ...
}
```

```java
public static void main(String[] args) {
... .. ...
}
```

```
E:\work\java\eclipse\HelloWorld\src>dir
 Volume in drive E has no label.
 Volume Serial Number is ####-####
\par
Directory of E:\work\java\eclipse\HelloWorld\src
.
..
179 helloworld.java
    p1
    p2
\par
1 File(s)              179 bytes
    4 Dir(s)  85,037,547,520 bytes free
```

```
// package p1
package p1;
\par
public class A {
  public void display(){
    System.out.println("Class A is called here!");
  }
```

```java
// package p2
package p2;
import p1.*;
\par
public class B {
\par
static void display(){
    System.out.println("This is class B!");
  }
  public static void main(String[] args){
    A obj = new A();
    obj.display();
    display();
  }
}
```

```java
//Java program to illustrate error while
//using class from different package with
//private modifier
package p1;
\par
class A
{
   private void display()
    {
        System.out.println("This is class A of p1");
    }
}
\par
class B
{
   public static void main(String args[])
      {
          A obj = new A();
          //trying to access private method of another class
          obj.display();
      }
}
```

```
//Java program to illustrate
//protected modifier
package p1;
\par
//Class A
public class A
{
   protected void display()
    {
        System.out.println("This is class A of p1");
    }
}
```

```java
//Java program to illustrate
//protected modifier
package p2;
import p1.*; //importing all classes in package p1
\par
//Class B is subclass of A
class B extends A
// class inheritance
{
   public static void main(String args[])
   {
       B obj = new B();
       obj.display();
   }
\par
}
```

```
package teststatic;
\par
public class TestStatic {
\par
static void callStatic(){
    System.out.println("Static method called in main method!");
  }
\par
void callNonStatic(){
    System.out.println("Non-static method called in main method!
    using 'instance'");
  }
\par
public static void main(String[] args) {
\par
// this static method
    callStatic();
\par
// this is non-staic method
    TestStatic obj = new TestStatic();
    obj.callNonStatic();
    // this is also possible
    obj.callStatic();
  }
}
```

```
package teststatic;
\par
class Parent {
    void show()
    {
        System.out.println("Parent");
    }
}
\par
// Parent inherit in Child class
class Child extends Parent {
\par
// override show() of Parent
    void show()
    {
        System.out.println("Child");
    }
}
\par
class TestStatic {
    public static void main(String[] args)
    {
        Parent p = new Parent();
        // calling Parent's show()
        p.show();
\par
Parent c = new Child();
        // calling Child's show()
        c.show();
    }
}
```

```java
package teststatic;
\par
class Parent {
    static void show()
    {
        System.out.println("Parent");
    }
}
\par
// Parent inherit in Child class
class Child extends Parent {
\par
// override show() of Parent
    void show()
    {
        System.out.println("Child");
    }
}
\par
class TestStatic {
    public static void main(String[] args)
    {
        Parent p = new Parent();
        // calling Parent's show()
        p.show();
\par
Parent c = new Child();
        // calling Child's show()
        c.show();
    }
}
```

```
Multiple markers at this line
  - This instance method cannot override the static method
   from Parent
```

```
int score;
```

int

score

`int`, `for`, `class`

true, false

null

```
int float;
```

float

```
int speedLimit = 80;
```

speedLimit

```
int speedLimit;
speedLimit = 80;
```

```
int speedLimit = 80;
... ... ...
speedLimit = 90;
```

```
int speedLimit = 80;
... ... ...
float speedLimit;
```

```
int speed;
```

speed

boolean

false

```java
class BooleanExample {
    public static void main(String[] args) {
\par
boolean flag = true;
        System.out.println(flag);
    }
}
```

byte

-128

$$[-128, 127]$$

```
class ByteExample {
    public static void main(String[] args) {
\par
byte range;
        range = 124;
        System.out.println(range);
    }
}
```

short

$$-32768 to 32767$$

$$[-32768, 32767]$$

```
class ShortExample {
    public static void main(String[] args) {
\par
short temperature;
        temperature = -200;
        System.out.println(temperature);
    }
}
```

$$-2^3 1$$

$$2^{31} - 1$$

$$2^{32} - 1$$

```
class IntExample {
    public static void main(String[] args) {
\par
int range = -4250000;
        System.out.println(range);
    }
}
```

long

$-2^6 3$

$$2^{63} - 1$$

$$2^{64} - 1$$

```java
class LongExample {
    public static void main(String[] args) {
\par
long range = -42332200000L;
        System.out.println(range);
    }
}
```

$$-42332200000$$

double

```java
class DoubleExample {
    public static void main(String[] args) {
\par
double number = -42.3;
        System.out.println(number);
    }
}
```

```
class FloatExample {
    public static void main(String[] args) {
\par
float number = -42.3f;
        System.out.println(number);
    }
}
\par
```

$$-42.3f$$

-42.3

$$'\backslash u0000'$$

$'\backslash uffff'$

```
class CharExample {
    public static void main(String[] args) {
\par
char letter = '\u0051';
        System.out.println(letter);
    }
}
```

$'\backslash u0051'$

```
class CharExample {
    public static void main(String[] args) {
\par
char letter1 = '9';
        System.out.println(letter1);
\par
char letter2 = 65;
        System.out.println(letter2);
\par
}
}
```

letter1

letter2

```
myString = "Programming is awesome";
```

```
boolean flag = false;
```

flag

$$1.5, 4, true, '\backslash u0050'$$

$-5,' a', true$

```java
// Error! literal 42332200000 of type int is out of range
int myVariable1 = 42332200000;// 42332200000L is of type long, and
    it's not out of range
long myVariable2 = 42332200000L;
```

$0x$

$0b$

```
// decimal
int decNumber = 34;
int hexNumber = 0x2F; // 0x represents hexadecimal
int binNumber = 0b10010; // 0b represents binary
```

```
class DoubleExample {
    public static void main(String[] args) {
\par
double myDouble = 3.4;
        float myFloat = 3.4F;
\par
// 3.445*10^2
        double myDoubleScientific = 3.445e2;
\par
System.out.println(myDouble);
        System.out.println(myFloat);
        System.out.println(myDoubleScientific);
    }
}
```

char

$'a', '\u0111'$

$\backslash b$

$\backslash t$

$\backslash n$

$\backslash f$

$\backslash r$

\ ”

\\

```java
class DoubleExample {
    public static void main(String[] args) {
\par
char myChar = 'g';
        char newLine = '\n';
        String myString = "Java 8";
\par
System.out.println(myChar);
        System.out.println(newLine);
        System.out.println(myString);
    }
}
```

```
int age;
age = 5;
```

```java
class AssignmentOperator {
    public static void main(String[] args) {
\par
int number1, number2;
\par
// Assigning 5 to number1
        number1 = 5;
        System.out.println(number1);
\par
// Assigning value of variable number2 to number1
        number2 = number1;
        System.out.println(number2);
    }
}
```

```java
class ArithmeticOperator {
    public static void main(String[] args) {
\par
double number1 = 12.5, number2 = 3.5, result;
\par
// Using addition operator
        result = number1 + number2;
        System.out.println("number1 + number2 = " + result);
\par
// Using subtraction operator
        result = number1 - number2;
        System.out.println("number1 - number2 = " + result);
\par
// Using multiplication operator
        result = number1 * number2;
        System.out.println("number1 * number2 = " + result);
\par
// Using division operator
        result = number1 / number2;
        System.out.println("number1 / number2 = " + result);
\par
// Using remainder operator
        result = number1 % number2;
        System.out.println("number1 % number2 = " + result);
    }
}
```

```
result = number1 + 5.2;
result = 2.3 + 4.5;
number2 = number1 -2.9;
```

```java
class ArithmeticOperator {
    public static void main(String[] args) {
\par
String start, middle, end, result;
\par
start = "Talk is cheap. ";
        middle = "Show me the code. ";
        end = "- Linus Torvalds";
\par
result = start + middle + end;
        System.out.println(result);
    }
}
```

```java
class UnaryOperator {
    public static void main(String[] args) {
\par
double number = 5.2, resultNumber;
        boolean flag = false;
\par
System.out.println("+number = " + +number);
        // number is equal to 5.2 here.
\par
System.out.println("-number = " + -number);
        // number is equal to 5.2 here.
\par
// ++number is equivalent to number = number + 1
        System.out.println("number = " + ++number);
        // number is equal to 6.2 here.
\par
// -- number is equivalent to number = number - 1
        System.out.println("number = " + --number);
        // number is equal to 5.2 here.
\par
System.out.println("!flag = " + !flag);
        // flag is still false.
    }
}
```

```
+number = 5.2
-number = -5.2
number = 6.2
number = 5.2
!flag = true
```

++

— —

```
int myInt = 5;
++myInt    // myInt becomes 6
myInt++    // myInt becomes 7
--myInt    // myInt becomes 6
myInt--    // myInt becomes 5
```

```
class UnaryOperator {
    public static void main(String[] args) {
\par
double number = 5.2;
\par
System.out.println(number++);
        System.out.println(number);
\par
System.out.println(++number);
        System.out.println(number);
    }
}
```

```
5.2
6.2
7.2
7.2
```

```
System.out.println(number++);
```

```
System.out.println(number);
```

```
System.out.println(++number);
```

```java
class RelationalOperator {
    public static void main(String[] args) {
\par
int number1 = 5, number2 = 6;
\par
if (number1 > number2) {
            System.out.println("number1 is greater than number2.");
        }
        else {
            System.out.println("number2 is greater than number1.");
        }
    }
}
```

```
class instanceofOperator {
    public static void main(String[] args) {
\par
String test = "asdf";
        boolean result;
\par
result = test instanceof String;
        System.out.println("Is test an object of String? " + result)
    ;
    }
}
```

```java
result = objectName instanceof className;
```

```java
class Main {
    public static void main (String[] args) {
        String name = "Programiz";
        Integer age = 22;
\par
System.out.println("Is name an instance of String: "+ (name
    instanceof String));
        System.out.println("Is age an instance of Integer: "+ (age
    instanceof Integer));
    }
}
```

```
Is name an instance of String: true
Is age an instance of Integer: true
```

```java
String[] array = new String[100];
```

```
dataType[] arrayName;
```

```
double[] data;
```

```
// declare an array
double[] data;
\par
// allocate memory
data = new Double[10];
```

```java
double[] data = new double[10];
```

```java
//declare and initialize and array
int[] age = {12, 4, 5, 2, 5};
```

```
// declare an array
int[] age = new int[5];
\par
// initialize array
age[0] = 12;
age[1] = 4;
age[2] = 5;
..
```

```
// access array elements
array[index]
```

```java
class Main {
 public static void main(String[] args) {
\par
// create an array
   int[] age = {12, 4, 5, 2, 5};
\par
// access each array elements
   System.out.println("Accessing Elements of Array:");
   System.out.println("First Element: " + age[0]);
   System.out.println("Second Element: " + age[1]);
   System.out.println("Third Element: " + age[2]);
   System.out.println("Fourth Element: " + age[3]);
   System.out.println("Fifth Element: " + age[4]);
 }
}
```

```
Accessing Elements of Array:
First Element: 12
Second Element: 4
Third Element: 5
Fourth Element: 2
Fifth Element: 5
```

```
class Main {
 public static void main(String[] args) {
\par
// create an array
   int[] age = {12, 4, 5};
\par
// loop through the array
   // using for loop
   System.out.println("Using for Loop:");
   for(int i = 0; i < age.length; i++) {
     System.out.println(age[i]);
   }
 }
}
```

```
Using for Loop:
12
4
5
```

```
class Main {
 public static void main(String[] args) {
\par
// create an array
   int[] age = {12, 4, 5};
\par
// loop through the array
   // using for loop
   System.out.println("Using for-each Loop:");
   for(int a : age) {
     System.out.println(a);
   }
 }
}
```

```
Using for-each Loop:
12
4
5
```

```
class Main {
 public static void main(String[] args) {
\par
int[] numbers = {2, -9, 0, 5, 12, -25, 22, 9, 8, 12};
   int sum = 0;
   Double average;
\par
// access all elements using for each loop
   // add each element in sum
   for (int number: numbers) {
     sum += number;
   }
\par
// get the total number of elements
   int arrayLength = numbers.length;
\par
// calculate the average
   // convert the average from int to double
   average =  ((double)sum / (double)arrayLength);
\par
System.out.println("Sum = " + sum);
   System.out.println("Average = " + average);
 }
}
```

```
Sum = 36
Average = 3.6
```

```
average = ((double)sum / (double)arrayLength);
```

```java
double[][] matrix = {{1.2, 4.3, 4.0},
       {4.1, -1.1}
};
```

```java
int[][] a = new int[3][4];
```

```java
String[][][] data = new String[3][4][2];
```

```java
int[][] a = {
       {1, 2, 3},
       {4, 5, 6, 9},
       {7},
};
```

```
class MultidimensionalArray {
    public static void main(String[] args) {
\par
// create a 2d array
        int[][] a = {
            {1, 2, 3},
            {4, 5, 6, 9},
            {7},
        };
\par
// calculate the length of each row
        System.out.println("Length of row 1: " + a[0].length);
        System.out.println("Length of row 2: " + a[1].length);
        System.out.println("Length of row 3: " + a[2].length);
    }
}
```

```
Length of row 1: 3
Length of row 2: 4
Length of row 3: 1
```

```
class MultidimensionalArray {
    public static void main(String[] args) {
\par
int[][] a = {
            {1, -2, 3},
            {-4, -5, 6, 9},
            {7},
        };
\par
for (int i = 0; i < a.length; ++i) {
            for(int j = 0; j < a[i].length; ++j) {
                System.out.println(a[i][j]);
            }
        }
    }
}
```

```
1
-2
3
-4
-5
6
9
7
```

```
class MultidimensionalArray {
    public static void main(String[] args) {
\par
// create a 2d array
        int[][] a = {
            {1, -2, 3},
            {-4, -5, 6, 9},
            {7},
        };
\par
// first for...each loop access the individual array
        // inside the 2d array
        for (int[] innerArray: a) {
            // second for...each loop access each element inside the
     row
            for(int data: innerArray) {
                System.out.println(data);
            }
        }
    }
}
```

```java
// test is a 3d array
int[][][] test = {
        {
          {1, -2, 3},
          {2, 3, 4}
        },
        {
          {-4, -5, 6, 9},
          {1},
          {2, 3}
        }
};
```

```
class ThreeArray {
    public static void main(String[] args) {
\par
// create a 3d array
        int[][][] test = {
            {
              {1, -2, 3},
              {2, 3, 4}
            },
            {
              {-4, -5, 6, 9},
              {1},
              {2, 3}
            }
        };
\par
// for..each loop to iterate through elements of 3d array
        for (int[][] array2D: test) {
            for (int[] array1D: array2D) {
                for(int item: array1D) {
                    System.out.println(item);
                }
            }
        }
    }
}
```

```
1
-2
3
2
3
4
-4
-5
6
9
1
2
3
```

```
class Main {
    public static void main(String[] args) {
\par
int [] numbers = {1, 2, 3, 4, 5, 6};
        int [] positiveNumbers = numbers;     // copying arrays
\par
for (int number: positiveNumbers) {
            System.out.print(number + ", ");
        }
    }
}
```

```
class Main {
    public static void main(String[] args) {
\par
int [] numbers = {1, 2, 3, 4, 5, 6};
        int [] positiveNumbers = numbers;     // copying arrays
\par
// change value of first array
        numbers[0] = -1;
\par
// printing the second array
        for (int number: positiveNumbers) {
            System.out.print(number + ", ");
        }
    }
}
```

```java
import java.util.Arrays;
\par
class Main {
    public static void main(String[] args) {
\par
int [] source = {1, 2, 3, 4, 5, 6};
        int [] destination = new int[6];
\par
// iterate and copy elements from source to destination
        for (int i = 0; i < source.length; ++i) {
            destination[i] = source[i];
        }
\par
// converting array to string
        System.out.println(Arrays.toString(destination));
    }
}
```

```
arraycopy(Object src, int srcPos,Object dest, int destPos, int
    length)
```

```java
// To use Arrays.toString() method
import java.util.Arrays;
\par
class Main {
    public static void main(String[] args) {
        int[] n1 = {2, 3, 12, 4, 12, -2};
\par
int[] n3 = new int[5];
\par
// Creating n2 array of having length of n1 array
        int[] n2 = new int[n1.length];
\par
// copying entire n1 array to n2
        System.arraycopy(n1, 0, n2, 0, n1.length);
        System.out.println("n2 = " + Arrays.toString(n2));
\par
// copying elements from index 2 on n1 array
        // copying element to index 1 of n3 array
        // 2 elements will be copied
        System.arraycopy(n1, 2, n3, 1, 2);
        System.out.println("n3 = " + Arrays.toString(n3));
    }
}
```

```
n2 = [2, 3, 12, 4, 12, -2]
n3 = [0, 12, 4, 0, 0]
```

```java
// To use toString() and copyOfRange() method
import java.util.Arrays;
\par
class ArraysCopy {
    public static void main(String[] args) {
\par
int[] source = {2, 3, 12, 4, 12, -2};
\par
// copying entire source array to destination
        int[] destination1 = Arrays.copyOfRange(source, 0, source.
    length);
        System.out.println("destination1 = " + Arrays.toString(
    destination1));
\par
// copying from index 2 to 5 (5 is not included)
        int[] destination2 = Arrays.copyOfRange(source, 2, 5);
        System.out.println("destination2 = " + Arrays.toString(
    destination2));
    }
}
```

```
destination1 = [2, 3, 12, 4, 12, -2]
destination2 = [12, 4, 12]
```

```java
int[] destination1 = Arrays.copyOfRange(source, 0, source.length);
```

```java
import java.util.Arrays;
\par
class Main {
    public static void main(String[] args) {
\par
int[][] source = {
              {1, 2, 3, 4},
              {5, 6},
              {0, 2, 42, -4, 5}
              };
\par
int[][] destination = new int[source.length][];
\par
for (int i = 0; i < destination.length; ++i) {
\par
// allocating space for each row of destination array
            destination[i] = new int[source[i].length];
\par
for (int j = 0; j < destination[i].length; ++j) {
                destination[i][j] = source[i][j];
            }
        }
\par
// displaying destination array
        System.out.println(Arrays.deepToString(destination));
\par
}
}
```

```
[[1, 2, 3, 4], [5, 6], [0, 2, 42, -4, 5]]
```

```
System.out.println(Arrays.deepToString(destination);
```

```java
import java.util.Arrays;
\par
class Main {
    public static void main(String[] args) {
\par
int[][] source = {
              {1, 2, 3, 4},
              {5, 6},
              {0, 2, 42, -4, 5}
              };
\par
int[][] destination = new int[source.length][];
\par
for (int i = 0; i < source.length; ++i) {
\par
// allocating space for each row of destination array
             destination[i] = new int[source[i].length];
             System.arraycopy(source[i], 0, destination[i], 0,
    destination[i].length);
        }
\par
// displaying destination array
        System.out.println(Arrays.deepToString(destination));
    }
}
```

```java
// The Collections framework is defined in the java.util package
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args){
        ArrayList<String> animals = new ArrayList<>();
        // Add elements
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Horse");
\par
System.out.println("ArrayList: " + animals);
    }
}
```

```java
// ArrayList implementation of List
List<String> list1 = new ArrayList<>();
\par
// LinkedList implementation of List
List<String> list2 = new LinkedList<>();
```

```java
import java.util.List;
import java.util.ArrayList;
\par
class Main {
\par
public static void main(String[] args) {
        // Creating list using the ArrayList class
        List<Integer> numbers = new ArrayList<>();
\par
// Add elements to the list
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        System.out.println("List: " + numbers);
\par
// Access element from the list
        int number = numbers.get(2);
        System.out.println("Accessed Element: " + number);
\par
// Remove element from the list
        int removedNumber = numbers.remove(1);
        System.out.println("Removed Element: " + removedNumber);
    }
}
```

```
List: [1, 2, 3]
Accessed Element: 3
Removed Element: 2
```

```
import java.util.List;
import java.util.LinkedList;
\par
class Main {
\par
public static void main(String[] args) {
        // Creating list using the LinkedList class
        List<Integer> numbers = new LinkedList<>();
\par
// Add elements to the list
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        System.out.println("List: " + numbers);
\par
// Access element from the list
        int number = numbers.get(2);
        System.out.println("Accessed Element: " + number);
\par
// Using the indexOf() method
        int index = numbers.indexOf(2);
        System.out.println("Position of 3 is " + index);
\par
// Remove element from the list
        int removedNumber = numbers.remove(1);
        System.out.println("Removed Element: " + removedNumber);
    }
}
```

```
List: [1, 2, 3]
Accessed Element: 3
Position of 3 is 1
Removed Element: 2
```

```
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args){
        ArrayList<String> animals = new ArrayList<>();
\par
// Add elements
        animals.add(0,"Dog");
        animals.add(1,"Cat");
        animals.add(2,"Horse");
        System.out.println("ArrayList: " + animals);
    }
}
```

```
ArrayList: [Dog, Cat, Horse]
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args){
        ArrayList<String> mammals = new ArrayList<>();
        mammals.add("Dog");
        mammals.add("Cat");
        mammals.add("Horse");
        System.out.println("Mammals: " + mammals);
\par
ArrayList<String> animals = new ArrayList<>();
        animals.add("Crocodile");
\par
// Add all elements of mammals in animals
        animals.addAll(mammals);
        System.out.println("Animals: " + animals);
    }
}
```

```
Mammals: [Dog, Cat, Horse]
Animals: [Crocodile, Dog, Cat, Horse]
```

```java
import java.util.ArrayList;
import java.util.Arrays;
\par
class Main {
    public static void main(String[] args) {
        // Creating an array list
        ArrayList<String> animals = new ArrayList<>(Arrays.asList("
    Cat", "Cow", "Dog"));
        System.out.println("ArrayList: " + animals);
\par
// Access elements of the array list
        String element = animals.get(1);
        System.out.println("Accessed Element: " + element);
    }
}
```

```
ArrayList: [Cat, Cow, Dog]
Accessed Elemenet: Cow
```

```
new ArrayList<>(Arrays.asList(("Cat", "Cow", "Dog"));
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        ArrayList<String> animals= new ArrayList<>();
\par
// Add elements in the array list
        animals.add("Dog");
        animals.add("Horse");
        animals.add("Cat");
        System.out.println("ArrayList: " + animals);
\par
// Get the element from the array list
        String str = animals.get(0);
        System.out.print("Element at index 0: " + str);
    }
}
```

```
ArrayList: [Dog, Horse, Cat]
Element at index 0: Dog
```

```java
import java.util.ArrayList;
import java.util.Iterator;
\par
class Main {
    public static void main(String[] args){
        ArrayList<String> animals = new ArrayList<>();
\par
// Add elements in the array list
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Horse");
        animals.add("Zebra");
\par
// Create an object of Iterator
        Iterator<String> iterate = animals.iterator();
        System.out.print("ArrayList: ");
\par
// Use methods of Iterator to access elements
        while(iterate.hasNext()){
            System.out.print(iterate.next());
            System.out.print(", ");
        }
    }
}
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        ArrayList<String> animals= new ArrayList<>();
        // Add elements in the array list
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Horse");
        System.out.println("ArrayList: " + animals);
\par
// Change the element of the array list
        animals.set(2, "Zebra");
        System.out.println("Modified ArrayList: " + animals);
    }
}
```

```
ArrayList: [Dog, Cat, Horse]
Modified ArrayList: [Dog, Cat, Zebra]
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        ArrayList<String> animals = new ArrayList<>();
\par
// Add elements in the array list
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Horse");
        System.out.println("Initial ArrayList: " + animals);
\par
// Remove element from index 2
        String str = animals.remove(2);
        System.out.println("Final ArrayList: " + animals);
        System. out.println("Removed Element: " + str);
    }
}
```

```
Initial ArrayList: [Dog, Cat, Horse]
Final ArrayList: [Dog, Cat]
Removed Element: Horse
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        ArrayList<String> animals = new ArrayList<>();
\par
// Add elements in the ArrayList
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Horse");
        System.out.println("Initial ArrayList: " + animals);
\par
// Remove all the elements
        animals.removeAll(animals);
        System.out.println("Final ArrayList: " + animals);
    }
}
```

```
Initial ArrayList: [Dog, Cat, Horse]
Final ArrayList: []
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        ArrayList<String> animals= new ArrayList<>();
\par
// Add elements in the array list
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Horse");
        System.out.println("Initial ArrayList: " + animals);
\par
// Remove all the elements
        animals.clear();
        System.out.println("Final ArrayList: " + animals);
    }
}
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        // Creating an array list
        ArrayList<String> animals = new ArrayList<>();
        animals.add("Cow");
        animals.add("Cat");
        animals.add("Dog");
        System.out.println("ArrayList: " + animals);
\par
// Using for loop
        System.out.println("Accessing individual elements: ");
\par
for(int i = 0; i < animals.size(); i++) {
            System.out.print(animals.get(i));
            System.out.print(", ");
        }
    }
}
```

```
ArrayList: [Cow, Cat, Dog]
Accessing individual elements:
Cow, Cat, Dog,
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        // Creating an array list
        ArrayList<String> animals = new ArrayList<>();
        animals.add("Cow");
        animals.add("Cat");
        animals.add("Dog");
        System.out.println("ArrayList: " + animals);
\par
// Using forEach loop
        System.out.println("Accessing individual elements:  ");
        for(String animal : animals) {
            System.out.print(animal);
            System.out.print(", ");
        }
    }
}
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        ArrayList<String> animals= new ArrayList<>();
\par
// Adding elements in the arrayList
        animals.add("Dog");
        animals.add("Horse");
        animals.add("Cat");
        System.out.println("ArrayList: " + animals);
\par
// getting the size of the arrayList
        System.out.println("Size: " + animals.size());
    }
}
```

```
ArrayList: [Dog, Horse, Cat]
Size: 3
```

```
import java.util.ArrayList;
import java.util.Collections;
\par
class Main {
    public static void main(String[] args){
        ArrayList<String> animals= new ArrayList<>();
\par
// Add elements in the array list
        animals.add("Horse");
        animals.add("Zebra");
        animals.add("Dog");
        animals.add("Cat");
\par
System.out.println("Unsorted ArrayList: " + animals);
\par
// Sort the array list
        Collections.sort(animals);
        System.out.println("Sorted ArrayList: " + animals);
    }
}
```

```
Unsorted ArrayList: [Horse, Zebra, Dog, Cat]
Sorted ArrayList: [Cat, Dog, Horse, Zebra]
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        ArrayList<String> animals= new ArrayList<>();
\par
// Add elements in the array list
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Horse");
        System.out.println("ArrayList: " + animals);
\par
// Create a new array of String type
        String[] arr = new String[animals.size()];
\par
// Convert ArrayList into an array
        animals.toArray(arr);
        System.out.print("Array: ");
        for(String item:arr) {
            System.out.print(item+", ");
        }
    }
}
```

```
ArrayList: [Dog, Cat, Horse]
Array: Dog, Cat, Horse,
```

```java
import java.util.ArrayList;
import java.util.Arrays;
\par
class Main {
    public static void main(String[] args) {
        // Create an array of String type
        String[] arr = {"Dog", "Cat", "Horse"};
        System.out.print("Array: ");
\par
// Print array
        for(String str: arr) {
            System.out.print(str);
            System.out.print(" ");
        }
\par
// Create an ArrayList from an array
        ArrayList<String> animals = new ArrayList<>(Arrays.asList(
    arr));
        System.out.println("\nArrayList: " + animals);
    }
}
```

```
Array: Dog, Cat, Horse
ArrayList: [Dog, Cat, Horse]
\par
```

```java
import java.util.ArrayList;
\par
class Main {
    public static void main(String[] args) {
        ArrayList<String> animals = new ArrayList<>();
\par
// Add elements in the ArrayList
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Horse");
        System.out.println("ArrayList: " + animals);
\par
// Convert ArrayList into an String
        String str = animals.toString();
        System.out.println("String: " + str);
    }
}
```

```
ArrayList: [Dog, Cat, Horse]
String: [Dog, Cat, Horse]
```

```java
Vector<Type> vector = new Vector<>();
```

```
// create Integer type linked list
Vector<Integer> vector= new Vector<>();
\par
// create String type linked list
Vector<String> vector= new Vector<>();
```

```
import java.util.Vector;
\par
class Main {
    public static void main(String[] args) {
        Vector<String> mammals= new Vector<>();
\par
// Using the add() method
        mammals.add("Dog");
        mammals.add("Horse");
\par
// Using index number
        mammals.add(2, "Cat");
        System.out.println("Vector: " + mammals);
\par
// Using addAll()
        Vector<String> animals = new Vector<>();
        animals.add("Crocodile");
\par
animals.addAll(mammals);
        System.out.println("New Vector: " + animals);
    }
}
```

```
Vector: [Dog, Horse, Cat]
New Vector: [Crocodile, Dog, Horse, Cat]
```

```java
import java.util.Iterator;
import java.util.Vector;
\par
class Main {
    public static void main(String[] args) {
        Vector<String> animals= new Vector<>();
        animals.add("Dog");
        animals.add("Horse");
        animals.add("Cat");
\par
// Using get()
        String element = animals.get(2);
        System.out.println("Element at index 2: " + element);
\par
// Using iterator()
        Iterator<String> iterate = animals.iterator();
        System.out.print("Vector: ");
        while(iterate.hasNext()) {
            System.out.print(iterate.next());
            System.out.print(", ");
        }
    }
}
```

```
Element at index 2: Cat
Vector: Dog, Horse, Cat,
```

```java
import java.util.Vector;
\par
class Main {
    public static void main(String[] args) {
        Vector<String> animals= new Vector<>();
        animals.add("Dog");
        animals.add("Horse");
        animals.add("Cat");
\par
System.out.println("Initial Vector: " + animals);
\par
// Using remove()
        String element = animals.remove(1);
        System.out.println("Removed Element: " + element);
        System.out.println("New Vector: " + animals);
\par
// Using clear()
        animals.clear();
        System.out.println("Vector after clear(): " + animals);
    }
}
```

```
Initial Vector: [Dog, Horse, Cat]
Removed Element: Horse
New Vector: [Dog, Cat]
Vector after clear(): []
```

```
Stack<Type> stacks = new Stack<>();
```

```
// Create Integer type stack
Stack<Integer> stacks = new Stack<>();
\par
// Create String type stack
Stack<String> stacks = new Stack<>();
```

```java
import java.util.Stack;
\par
class Main {
    public static void main(String[] args) {
        Stack<String> animals= new Stack<>();
\par
// Add elements to Stack
        animals.push("Dog");
        animals.push("Horse");
        animals.push("Cat");
\par
System.out.println("Stack: " + animals);
    }
}
```

```
Stack: [Dog, Horse, Cat]
```

```java
import java.util.Stack;
\par
class Main {
    public static void main(String[] args) {
        Stack<String> animals= new Stack<>();
\par
// Add elements to Stack
        animals.push("Dog");
        animals.push("Horse");
        animals.push("Cat");
        System.out.println("Initial Stack: " + animals);
\par
// Remove element stacks
        String element = animals.pop();
        System.out.println("Removed Element: " + element);
    }
}
```

```
import java.util.Stack;
\par
class Main {
    public static void main(String[] args) {
        Stack<String> animals= new Stack<>();
\par
// Add elements to Stack
        animals.push("Dog");
        animals.push("Horse");
        animals.push("Cat");
        System.out.println("Stack: " + animals);
\par
// Access element from the top
        String element = animals.peek();
        System.out.println("Element at top: " + element);
\par
}
}
```

```
Stack: [Dog, Horse, Cat]
Element at top: Cat
```

```java
import java.util.Stack;
\par
class Main {
    public static void main(String[] args) {
        Stack<String> animals= new Stack<>();
\par
// Add elements to Stack
        animals.push("Dog");
        animals.push("Horse");
        animals.push("Cat");
        System.out.println("Stack: " + animals);
\par
// Search an element
        int position = animals.search("Horse");
        System.out.println("Position of Horse: " + position);
    }
}
```

```
Stack: [Dog, Horse, Cat]
Position of Horse: 2
```

```java
import java.util.Stack;
\par
class Main {
    public static void main(String[] args) {
        Stack<String> animals= new Stack<>();
\par
// Add elements to Stack
        animals.push("Dog");
        animals.push("Horse");
        animals.push("Cat");
        System.out.println("Stack: " + animals);
\par
// Check if stack is empty
        boolean result = animals.empty();
        System.out.println("Is the stack empty? " + result);
    }
}
```

```
Stack: [Dog, Horse, Cat]
Is the stack empty? false
```

```java
// LinkedList implementation of Queue
Queue<String> animal1 = new LinkedList<>();
\par
// Array implementation of Queue
Queue<String> animal2 = new ArrayDeque<>();
\par
// Priority Queue implementation of Queue
Queue<String> animal3 = new PriorityQueue<>();
\par
```

```java
import java.util.Queue;
import java.util.LinkedList;
\par
class Main {
\par
public static void main(String[] args) {
        // Creating Queue using the LinkedList class
        Queue<Integer> numbers = new LinkedList<>();
\par
// offer elements to the Queue
        numbers.offer(1);
        numbers.offer(2);
        numbers.offer(3);
        System.out.println("Queue: " + numbers);
\par
// Access elements of the Queue
        int accessedNumber = numbers.peek();
        System.out.println("Accessed Element: " + accessedNumber);
\par
// Remove elements from the Queue
        int removedNumber = numbers.poll();
        System.out.println("Removed Element: " + removedNumber);
\par
System.out.println("Updated Queue: " + numbers);
    }
}
```

```
Queue: [1, 2, 3]
Accessed Element: 1
Removed Element: 1
Updated Queue: [2, 3]
```

```java
import java.util.PriorityQueue;
import java.util.Iterator;
\par
class Main {
    public static void main(String[] args) {
\par
// Creating a priority queue
        PriorityQueue<Integer> numbers = new PriorityQueue<>();
        numbers.add(4);
        numbers.add(2);
        numbers.add(1);
        System.out.print("PriorityQueue using iterator(): ");
\par
//Using the iterator() method
        Iterator<Integer> iterate = numbers.iterator();
        while(iterate.hasNext()) {
            System.out.print(iterate.next());
            System.out.print(", ");
        }
    }
}
```

```java
// Array implementation of Deque
Deque<String> animal1 = new ArrayDeque<>();
\par
// LinkedList implementation of Deque
Deque<String> animal2 = new LinkedList<>();
```

```
import java.util.Deque;
import java.util.ArrayDeque;
\par
class Main {
\par
public static void main(String[] args) {
        // Creating Deque using the ArrayDeque class
        Deque<Integer> numbers = new ArrayDeque<>();
\par
// add elements to the Deque
        numbers.offer(1);
        numbers.offerLast(2);
        numbers.offerFirst(3);
        System.out.println("Deque: " + numbers);
\par
// Access elements of the Deque
        int firstElement = numbers.peekFirst();
        System.out.println("First Element: " + firstElement);
\par
int lastElement = numbers.peekLast();
        System.out.println("Last Element: " + lastElement);
\par
// Remove elements from the Deque
        int removedNumber1 = numbers.pollFirst();
        System.out.println("Removed First Element: " +
    removedNumber1);
\par
int removedNumber2 = numbers.pollLast();
        System.out.println("Removed Last Element: " + removedNumber2
    );
\par
System.out.println("Updated Deque: " + numbers);
    }
}
```

```
Deque: [3, 1, 2]
First Element: 3
Last Element: 2
Removed First Element: 3
Removed Last Element: 2
Updated Deque: [1]
\par
```

```java
// Map implementation using HashMap
Map<Key, Value> numbers = new HashMap<>();
\par
```

```java
import java.util.Map;
import java.util.HashMap;
\par
class Main {
\par
public static void main(String[] args) {
        // Creating a map using the HashMap
        Map<String, Integer> numbers = new HashMap<>();
\par
// Insert elements to the map
        numbers.put("One", 1);
        numbers.put("Two", 2);
        System.out.println("Map: " + numbers);
\par
// Access keys of the map
        System.out.println("Keys: " + numbers.keySet());
\par
// Access values of the map
        System.out.println("Values: " + numbers.values());
\par
// Access entries of the map
        System.out.println("Entries: " + numbers.entrySet());
\par
// Remove Elements from the map
        int value = numbers.remove("Two");
        System.out.println("Removed Value: " + value);
    }
}
```

```
Map: {One=1, Two=2}
Keys: [One, Two]
Values: [1, 2]
Entries: [One=1, Two=2]
Removed Value: 2
```

```java
import java.util.Map;
import java.util.TreeMap;
\par
class Main {
\par
public static void main(String[] args) {
        // Creating Map using TreeMap
        Map<String, Integer> values = new TreeMap<>();
\par
// Insert elements to map
        values.put("Second", 2);
        values.put("First", 1);
        System.out.println("Map using TreeMap: " + values);
\par
// Replacing the values
        values.replace("First", 11);
        values.replace("Second", 22);
        System.out.println("New Map: " + values);
\par
// Remove elements from the map
        int removedValue = values.remove("First");
        System.out.println("Removed Value: " + removedValue);
    }
}
```

```
Map using TreeMap: {First=1, Second=2}
New Map: {First=11, Second=22}
Removed Value: 11
```

```
import java.util.HashMap;
\par
class Main {
    public static void main(String[] args) {
        // Creating HashMap of even numbers
        HashMap<String, Integer> evenNumbers = new HashMap<>();
\par
// Using put()
        evenNumbers.put("Two", 2);
        evenNumbers.put("Four", 4);
\par
// Using putIfAbsent()
        evenNumbers.putIfAbsent("Six", 6);
        System.out.println("HashMap of even numbers: " + evenNumbers
    );
\par
//Creating HashMap of numbers
        HashMap<String, Integer> numbers = new HashMap<>();
        numbers.put("One", 1);
\par
// Using putAll()
        numbers.putAll(evenNumbers);
        System.out.println("HashMap of numbers: " + numbers);
    }
}
```

```
HashMap of even numbers: {Six=6, Four=4, Two=2}
HashMap of numbers: {Six=6, One=1, Four=4, Two=2}
```

```
import java.util.HashMap;
\par
class Main {
    public static void main(String[] args) {
        HashMap<String, Integer> numbers = new HashMap<>();
\par
numbers.put("One", 1);
        numbers.put("Two", 2);
        numbers.put("Three", 3);
        System.out.println("HashMap: " + numbers);
\par
// Using entrySet()
        System.out.println("Key/Value mappings: " + numbers.entrySet
    ());
\par
// Using keySet()
        System.out.println("Keys: " + numbers.keySet());
\par
// Using values()
        System.out.println("Values: " + numbers.values());
    }
}
```

```
HashMap: {One=1, Two=2, Three=3}
Key/Value mappings: [One=1, Two=2, Three=3]
Keys: [One, Two, Three]
Values: [1, 2, 3]
```

```java
import java.util.HashMap;
\par
class Main {
    public static void main(String[] args) {
\par
HashMap<String, Integer> numbers = new HashMap<>();
        numbers.put("One", 1);
        numbers.put("Two", 2);
        numbers.put("Three", 3);
        System.out.println("HashMap: " + numbers);
\par
// Using get()
        int value1 = numbers.get("Three");
        System.out.println("Returned Number: " + value1);
\par
// Using getOrDefault()
        int value2 = numbers.getOrDefault("Five", 5);
        System.out.println("Returned Number: " + value2);
    }
}
```

```
HashMap: {One=1, Two=2, Three=3}
Returned Number: 3
Returned Number: 5
```

```
import java.util.HashMap;
\par
class Main {
    public static void main(String[] args) {
\par
HashMap<String, Integer> numbers = new HashMap<>();
        numbers.put("One", 1);
        numbers.put("Two", 2);
        numbers.put("Three", 3);
        System.out.println("HashMap: " + numbers);
\par
// remove method with single parameter
        int value = numbers.remove("Two");
        System.out.println("Removed value: " + value);
\par
// remove method with two parameters
        boolean result = numbers.remove("Three", 3);
        System.out.println("Is the entry Three removed? " + result);
\par
System.out.println("Updated HashMap: " + numbers);
    }
}
```

```
HashMap: {One=1, Two=2, Three=3}
Removed value: 2
Is the entry Three removed? True
Updated HashMap: {One=1}
```

```java
import java.util.HashMap;
\par
class Main {
    public static void main(String[] args) {
\par
HashMap<String, Integer> numbers = new HashMap<>();
        numbers.put("First", 1);
        numbers.put("Second", 2);
        numbers.put("Third", 3);
        System.out.println("Original HashMap: " + numbers);
\par
// Using replace()
        numbers.replace("Second", 22);
        numbers.replace("Third", 3, 33);
        System.out.println("HashMap using replace(): " + numbers);
\par
// Using replaceAll()
        numbers.replaceAll((key, oldValue) -> oldValue + 2);
        System.out.println("HashMap using replaceAll(): " + numbers)
    ;
    }
}
```

```
Original HashMap: {Second=2, Third=3, First=1}
HashMap using replace: {Second=22, Third=33, First=1}
HashMap using replaceAll: {Second=24, Third=35, First=3}
```

```
import java.util.HashMap;
\par
class Main {
    public static void main(String[] args) {
\par
HashMap<String, Integer> numbers = new HashMap<>();
        numbers.put("First", 1);
        numbers.put("Second", 2);
        System.out.println("Original HashMap: " + numbers);
\par
// Using compute()
        numbers.compute("First", (key, oldValue) -> oldValue + 2);
        numbers.compute("Second", (key, oldValue) -> oldValue + 1);
        System.out.println("HashMap using compute(): " + numbers);
\par
// Using computeIfAbsent()
        numbers.computeIfAbsent("Three", key -> 5);
        System.out.println("HashMap using computeIfAbsent(): " +
    numbers);
\par
// Using computeIfPresent()
        numbers.computeIfPresent("Second", (key, oldValue) ->
    oldValue * 2);
        System.out.println("HashMap using computeIfPresent(): " +
    numbers);
    }
}
\par
```

```
Original HashMap: {Second=2, First=1}
HashMap using compute(): {Second=3, First=3}
HashMap using computeIfAbsent(): {Second=3 First=3, Three=5}
HashMap using computeIfPresent(): {Second=6, First=3, three=5}
```

```
import java.util.HashMap;
\par
class Main {
    public static void main(String[] args) {
\par
HashMap<String, Integer> numbers = new HashMap<>();
        numbers.put("First", 1);
        numbers.put("Second", 2);
        System.out.println("Original HashMap: " + numbers);
\par
// Using merge() Method
        numbers.merge("First", 4, (oldValue, newValue) -> oldValue +
     newValue);
        System.out.println("New HashMap: " + numbers);
    }
}
\par
```

```
Original HashMap: {Second=2, First=1}
New HashMap: {Second=2, First=5}
```

```java
import java.util.HashMap;
import java.util.Map.Entry;
\par
class Main {
    public static void main(String[] args) {
\par
// Creating a HashMap
        HashMap<String, Integer> numbers = new HashMap<>();
        numbers.put("One", 1);
        numbers.put("Two", 2);
        numbers.put("Three", 3);
        System.out.println("HashMap: " + numbers);
\par
// Accessing the key/value pair
        System.out.print("Entries: ");
        for(Entry<String, Integer> entry: numbers.entrySet()) {
            System.out.print(entry);
            System.out.print(", ");
        }
\par
// Accessing the key
        System.out.print("\nKeys: ");
        for(String key: numbers.keySet()) {
            System.out.print(key);
            System.out.print(", ");
        }
\par
// Accessing the value
        System.out.print("\nValues: ");
        for(Integer value: numbers.values()) {
            System.out.print(value);
            System.out.print(", ");
        }
    }
}
```

```
HashMap: {One=1, Two=2, Three=3}
Entries: One=1, Two=2, Three=3
Keys: One, Two, Three,
Values: 1, 2, ,3,
```

```java
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map.Entry;
\par
class Main {
    public static void main(String[] args) {
        // Creating a HashMap
        HashMap<String, Integer> numbers = new HashMap<>();
        numbers.put("One", 1);
        numbers.put("Two", 2);
        numbers.put("Three", 3);
        System.out.println("HashMap: " + numbers);
\par
// Creating an object of Iterator
        Iterator<Entry<String, Integer>> iterate1 = numbers.entrySet
    ().iterator();
\par
// Accessing the Key/Value pair
        System.out.print("Entries: ");
        while(iterate1.hasNext()) {
            System.out.print(iterate1.next());
            System.out.print(", ");
        }
\par
// Accessing the key
        Iterator<String> iterate2 = numbers.keySet().iterator();
        System.out.print("\nKeys: ");
        while(iterate2.hasNext()) {
            System.out.print(iterate2.next());
            System.out.print(", ");
        }
\par
// Accessing the value
        Iterator<Integer> iterate3 = numbers.values().iterator();
         System.out.print("\nValues: ");
        while(iterate3.hasNext()) {
            System.out.print(iterate3.next());
            System.out.print(", ");
        }
    }
}
```

```
HashMap: {One=1, Two=2, Three=3}
Entries: One=1, Two=2, Three=3
Keys: One, Two, Three,
Values: 1, 2, 3,
```

```java
// Set implementation using HashSet
Set<String> animals = new HashSet<>();
```

```java
import java.util.Set;
import java.util.HashSet;
\par
class Main {
\par
public static void main(String[] args) {
        // Creating a set using the HashSet class
        Set<Integer> set1 = new HashSet<>();
\par
// Add elements to the set1
        set1.add(2);
        set1.add(3);
        System.out.println("Set1: " + set1);
\par
// Creating another set using the HashSet class
        Set<Integer> set2 = new HashSet<>();
\par
// Add elements
        set2.add(1);
        set2.add(2);
        System.out.println("Set2: " + set2);
\par
// Union of two sets
        set2.addAll(set1);
        System.out.println("Union is: " + set2);
    }
}
```

```
Set1: [2, 3]
Set2: [1, 2]
Union is: [1, 2, 3]
```

```
import java.util.Set;
import java.util.TreeSet;
import java.util.Iterator;
\par
class Main {
\par
public static void main(String[] args) {
        // Creating a set using the TreeSet class
        Set<Integer> numbers = new TreeSet<>();
\par
// Add elements to the set
        numbers.add(2);
        numbers.add(3);
        numbers.add(1);
        System.out.println("Set using TreeSet: " + numbers);
\par
// Access Elements using iterator()
        System.out.print("Accessing elements using iterator(): ");
        Iterator<Integer> iterate = numbers.iterator();
        while(iterate.hasNext()) {
            System.out.print(iterate.next());
            System.out.print(", ");
        }
\par
}
}
```

```
Set using TreeSet: [1, 2, 3]
Accessing elements using iterator(): 1, 2, 3,
```

```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

```java
// Creates an InputStream
InputStream object1 = new FileInputStream();
```

```
This is a line of text inside the file.
```

```
import java.io.FileInputStream;
import java.io.InputStream;
\par
public class Main {
    public static void main(String args[]) {
\par
byte[] array = new byte[100];
\par
try {
            InputStream input = new FileInputStream("input.txt");
\par
System.out.println("Available bytes in the file: " + input.available
    ());
\par
// Read byte from the input stream
            input.read(array);
            System.out.println("Data read from the file: ");
\par
// Convert byte array into string
            String data = new String(array);
            System.out.println(data);
\par
// Close the input stream
            input.close();
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Available bytes in the file: 35
Data read from the file:
This is a line of text inside the file
```

```java
// Creates an OutputStream
OutputStream object = new FileOutputStream();
```

```
import java.io.FileOutputStream;
import java.io.OutputStream;
\par
public class Main {
\par
public static void main(String args[]) {
        String data = "This is a line of text inside the file.";
\par
try {
            OutputStream out = new FileOutputStream("output.txt");
\par
// Converts the string into bytes
            byte[] dataBytes = data.getBytes();
\par
// Writes data to the output stream
            out.write(dataBytes);
            System.out.println("Data is written to the file.");
\par
// Closes the output stream
            out.close();
        }
\par
catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

```java
// Creates a Reader
Reader input = new FileReader();
```

```java
import java.io.Reader;
import java.io.FileReader;
\par
class Main {
    public static void main(String[] args) {
\par
// Creates an array of character
        char[] array = new char[100];
\par
try {
            // Creates a reader using the FileReader
            Reader input = new FileReader("input.txt");
\par
// Checks if reader is ready
            System.out.println("Is there data in the stream?  " +
    input.ready());
\par
// Reads characters
            input.read(array);
            System.out.println("Data in the stream:");
            System.out.println(array);
\par
// Closes the reader
            input.close();
        }
\par
catch(Exception e) {
            e.getStackTrace();
        }
    }
}
```

```
Is there data in the stream?  true
Data in the stream:
This is a line of text inside the file.
```

```java
// Creates a Writer
Writer output = new FileWriter();
```

```
import java.io.FileWriter;
import java.io.Writer;
\par
public class Main {
\par
public static void main(String args[]) {
\par
String data = "This is the data in the output file";
\par
try {
            // Creates a Writer using FileWriter
            Writer output = new FileWriter("output.txt");
\par
// Writes string to the file
            output.write(data);
\par
// Closes the writer
            output.close();
        }
\par
catch (Exception e) {
            e.getStackTrace();
        }
    }
}
\par
```

```
if (expression) {
    // statements
}
```

```java
class IfStatement {
    public static void main(String[] args) {
\par
int number = 10;
\par
// checks if number is greater than 0
        if (number > 0) {
            System.out.println("The number is positive.");
        }
        System.out.println("This statement is always executed.");
    }
}
```

```
The number is positive.
This statement is always executed.
```

```
class Main {
  public static void main(String[] args) {
    // create a string variable
    String language = "Java";
\par
// if statement
    if(language == "Java") {
      System.out.println("This is best programming language.");
    }
  }
}
```

```
if (expression) {
   // codes
}
else {
  // some other code
}
```

```
class IfElse {
    public static void main(String[] args) {
        int number = 10;
\par
// checks if number is greater than 0
        if (number > 0) {
            System.out.println("The number is positive.");
        }
        else {
            System.out.println("The number is not positive.");
        }
\par
System.out.println("This statement is always executed.");
    }
}
```

```
The number is not positive.
This statement is always executed.
```

```
if (expression1) {
   // codes
}
else if(expression2) {
   // codes
}
else if (expression3) {
   // codes
}
.
.
else {
   // codes
}
```

```
class Ladder {
    public static void main(String[] args) {
\par
int number = 0;
\par
// checks if number is greater than 0
        if (number > 0) {
            System.out.println("The number is positive.");
        }
\par
// checks if number is less than 0
        else if (number < 0) {
            System.out.println("The number is negative.");
        }
        else {
            System.out.println("The number is 0.");
        }
    }
}
```

```
class Number {
    public static void main(String[] args) {
\par
// declaring double type variables
        Double n1 = -1.0, n2 = 4.5, n3 = -5.3, largestNumber;
\par
// checks if n1 is greater than or equal to n2
        if (n1 >= n2) {
\par
// if...else statement inside the if block
            // checks if n1 is greater than or equal to n3
            if (n1 >= n3) {
                largestNumber = n1;
            }
\par
else {
                largestNumber = n3;
            }
        }
        else {
\par
// if..else statement inside else block
            // checks if n2 is greater than or equal to n3
            if (n2 >= n3) {
                largestNumber = n2;
            }
\par
else {
                largestNumber = n3;
            }
        }
\par
System.out.println("The largest number is " + largestNumber);
    }
}
```

```
if (expression) {
   number = 10;
}
else {
   number = -10;
}
```

```
number = (expression) ? expressionTrue : expressinFalse;
```

```
class Operator {
   public static void main(String[] args) {
\par
Double number = -5.5;
      String result;
\par
result = (number > 0.0) ? "positive" : "not positive";
      System.out.println(number + " is " + result);
   }
}
```

```
if (expression1) {
  result = 1;
} else if (expression2) {
  result = 2;
} else if (expression3) {
  result = 3;
} else {
  result = 0;
}
```

```
result = (expression1) ? 1 : (expression2) ? 2 : (expression3) ? 3 :
     0;
```

```
switch (variable/expression) {
case value1:
   // statements of case1
   break;
\par
case value2:
   // statements of case2
   break;
\par
.. .. ...
   .. .. ...
\par
default:
   // default statements
}
```

```java
class Main {
    public static void main(String[] args) {
\par
int week = 4;
        String day;
\par
// switch statement to check day
        switch (week) {
            case 1:
                day = "Sunday";
                break;
            case 2:
                day = "Monday";
                break;
            case 3:
                day = "Tuesday";
                break;
\par
// match the value of week
            case 4:
                day = "Wednesday";
                break;
            case 5:
                day = "Thursday";
                break;
            case 6:
                day = "Friday";
                break;
            case 7:
                day = "Saturday";
                break;
            default:
                day = "Invalid day";
                break;
        }
        System.out.println("The day is " + day);
    }
}
```

```
import java.util.Scanner;
\par
class Main {
    public static void main(String[] args) {
\par
char operator;
        Double number1, number2, result;
\par
// create an object of Scanner class
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter operator (either +, -, * or /): ");
\par
// ask user to enter operator
        operator = scanner.next().charAt(0);
        System.out.print("Enter number1 and number2 respectively: ")
    ;
\par
// ask user to enter numbers
        number1 = scanner.nextDouble();
        number2 = scanner.nextDouble();
\par
switch (operator) {
\par
// performs addition between numbers
            case '+':
                result = number1 + number2;
                System.out.print(number1 + "+" + number2 + " = " +
    result);
                break;
\par
// performs subtraction between numbers
            case '-':
                result = number1 - number2;
                System.out.print(number1 + "-" + number2 + " = " +
    result);
                break;
\par
// performs multiplication between numbers
            case '*':
                result = number1 * number2;
                System.out.print(number1 + "*" + number2 + " = " +
    result);
                break;
\par
// performs division between numbers
            case '/':
                result = number1 / number2;
                System.out.print(number1 + "/" + number2 + " = " +
    result);
                break;
\par
default:
                System.out.println("Invalid operator!");
                break;
        }
    }
}
```

```
Enter operator (either +, -, * or /): *
Enter number1 and number2 respectively: 1.4
-5.3
1.4*-5.3 = -7.419999999999999
```

```
for (initialization; testExpression; update)
{
    // codes inside for loop's body
}
```

```java
// Program to print a sentence 10 times
\par
class Loop {
    public static void main(String[] args) {
\par
for (int i = 1; i <= 10; ++i) {
          System.out.println("Line " + i);
      }
    }
}
```

```
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
```

```java
// Program to find the sum of natural numbers from 1 to 1000.
\par
class Number {
    public static void main(String[] args) {
\par
int sum = 0;
\par
for (int i = 1; i <= 1000; ++i) {
            sum += i;     // sum = sum + i
        }
\par
System.out.println("Sum = " + sum);
    }
}
```

```
// Infinite for Loop
\par
class Infinite {
    public static void main(String[] args) {
\par
int sum = 0;
\par
for (int i = 1; i <= 10; --i) {
            System.out.println("Hello");
        }
    }
}
```

```java
for (int a : array) {
    System.out.println(a);
}
```

```
for(data_type item : collections) {
    ...
}
```

```
// The program below calculates the sum of all elements of an
    integer array.
\par
class Main {
 public static void main(String[] args) {
\par
// create array
   int[] numbers = {3, 4, 5, -5, 0, 12};
   int sum = 0;
\par
// for each loop
   for (int number: numbers) {
     sum += number;
   }
\par
System.out.println("Sum = " + sum);
 }
}
```

```java
class Main {
 public static void main(String[] args) {
\par
char[] vowels = {'a', 'e', 'i', 'o', 'u'};
\par
// using for loop
   for (int i = 0; i < vowels.length; ++ i) {
     System.out.println(vowels[i]);
   }
 }
}
```

```
a
e
i
o
u
```

```java
class Main {
 public static void main(String[] args) {
\par
// create a char array
   char[] vowels = {'a', 'e', 'i', 'o', 'u'};
\par
// foreach loop
   for (char item: vowels) {
     System.out.println(item);
   }
 }
}
```

```
while (testExpression) {
    // codes inside the body of while loop
}
```

```
// Program to print line 10 times
\par
class Loop {
    public static void main(String[] args) {
\par
int i = 1;
\par
while (i <= 10) {
            System.out.println("Line " + i);
            ++i;
        }
    }
}
```

```
// Program to find the sum of natural numbers from 1 to 100.
\par
class AssignmentOperator {
    public static void main(String[] args) {
\par
int sum = 0, i = 100;
\par
while (i != 0) {
            sum += i;     // sum = sum + i;
            --i;
        }
\par
System.out.println("Sum = " + sum);
    }
}
```

```
do {
   // codes inside body of do while loop
} while (testExpression);
```

```
import java.util.Scanner;
\par
class Sum {
    public static void main(String[] args) {
\par
Double number, sum = 0.0;
        // creates an object of Scanner class
        Scanner input = new Scanner(System.in);
\par
do {
\par
// takes input from the user
            System.out.print("Enter a number: ");
            number = input.nextDouble();
            sum += number;
        } while (number != 0.0);  // test expression
\par
System.out.println("Sum = " + sum);
    }
}
```

```
Enter a number: 2.5
Enter a number: 23.3
Enter a number: -4.2
Enter a number: 3.4
Enter a number: 0
Sum = 25.0
```

```
// Infinite while loop
while (true) {
   // body of while loop
}
```

```java
// Infinite while loop
int i = 100;
while (i == 100) {
   System.out.print("Hey!");
}
```

```java
class Test {
    public static void main(String[] args) {
\par
// for loop
        for (int i = 1; i <= 10; ++i) {
\par
// if the value of i is 5 the loop terminates
            if (i == 5) {
                break;
            }
            System.out.println(i);
        }
    }
}
```

```
if (i == 5) {
    break;
}
```

```java
import java.util.Scanner;
\par
class UserInputSum {
    public static void main(String[] args) {
\par
Double number, sum = 0.0;
\par
// create an object of Scanner
        Scanner input = new Scanner(System.in);
\par
while (true) {
            System.out.print("Enter a number: ");
\par
// takes double input from user
            number = input.nextDouble();
\par
// if number is negative the loop terminates
            if (number < 0.0) {
                break;
            }
\par
sum += number;
        }
        System.out.println("Sum = " + sum);
    }
}
```

```
Enter a number: 3.2
Enter a number: 5
Enter a number: 2.3
Enter a number: 0
Enter a number: -4.5
Sum = 10.5
```

```
if (number < 0.0) {
    break;
}
```

```
while (testExpression) {
   // codes
   second:
   while (testExpression) {
      // codes
      while(testExpression) {
         // codes
         break second;
      }
   }
   // control jumps here
}
```

```
class LabeledBreak {
    public static void main(String[] args) {
\par
// the for loop is labeled as first
        first:
        for( int i = 1; i < 5; i++) {
\par
// the for loop is labeled as second
            second:
            for(int j = 1; j < 3; j ++ ) {
                System.out.println("i = " + i + "; j = " +j);
\par
// the break statement breaks the first for loop
                if ( i == 2)
                    break first;
            }
        }
    }
}
```

```
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1
```

```
first:
for(int i = 1; i < 5; i++) {...}
```

```
class LabeledBreak {
    public static void main(String[] args) {
\par
// the for loop is labeled as first
        first:
        for( int i = 1; i < 5; i++) {
\par
// the for loop is labeled as second
            second:
            for(int j = 1; j < 3; j ++ ) {
\par
System.out.println("i = " + i + "; j = " +j);
\par
// the break statement terminates the loop labeled as second
                if ( i == 2)
                    break second;
            }
        }
    }
}
```

```
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1
i = 3; j = 1
i = 3; j = 2
i = 4; j = 1
i = 4; j = 2
```

```java
class Test {
    public static void main(String[] args) {
\par
// for loop
        for (int i = 1; i <= 10; ++i) {
\par
// if value of i is between 4 and 9, continue is executed
            if (i > 4 && i < 9) {
                continue;
            }
            System.out.println(i);
        }
    }
}
```

```
1
2
        3
4
9
10
```

```
if (i > 5 && i < 9) {
    continue;
}
```

```
import java.util.Scanner;
\par
class AssignmentOperator {
    public static void main(String[] args) {
\par
Double number, sum = 0.0;
        // create an object of Scanner
        Scanner input = new Scanner(System.in);
\par
for (int i = 1; i < 6; ++i) {
            System.out.print("Enter a number: ");
            // takes double type input from the user
            number = input.nextDouble();
\par
// if number is negative, the iteration is skipped
            if (number <= 0.0) {
                continue;
            }
\par
sum += number;
        }
        System.out.println("Sum = " + sum);
    }
}
```

```
Enter a number: 2.2
Enter a number: 5.6
Enter a number: 0
Enter a number: -2.4
Enter a number: -3
Sum = 7.8
```

```java
class LabeledContinue {
    public static void main(String[] args) {
\par
// the outer for loop is labeled as label
        first:
        for (int i = 1; i < 6; ++i) {
            for (int j = 1; j < 5; ++j) {
                if (i == 3 || j == 2)
\par
// skips the iteration of label (outer for loop)
                    continue first;
                System.out.println("i = " + i + "; j = " + j);
            }
        }
    }
}
```

```
i = 1; j = 1
i = 2; j = 1
i = 4; j = 1
i = 5; j = 1
```

```
if (i==3 || j==2)
    continue first;
```

```
first:
for (int i = 1; i < 6; ++i) {..}
```

this

self

```
enum Size {
    constant1 , constant2 ,     , constantN;
\par
// methods and fields
}
```

```
enum Size {
   SMALL, MEDIUM, LARGE, EXTRALARGE
}
```

```java
enum Size {
   SMALL, MEDIUM, LARGE, EXTRALARGE
}
\par
class Main {
   public static void main(String[] args) {
      System.out.println(Size.SMALL);
      System.out.println(Size.MEDIUM);
   }
}
```

```
SMALL
MEDIUM
```

```
pizzaSize = Size.SMALL;
pizzaSize = Size.MEDIUM;
pizzaSize = Size.LARGE;
pizzaSize = Size.EXTRALARGE;
```

```
enum Size {
 SMALL, MEDIUM, LARGE, EXTRALARGE
}
\par
class Test {
 Size pizzaSize;
 public Test(Size pizzaSize) {
   this.pizzaSize = pizzaSize;
 }
 public void orderPizza() {
   switch(pizzaSize) {
     case SMALL:
       System.out.println("I ordered a small size pizza.");
       break;
     case MEDIUM:
       System.out.println("I ordered a medium size pizza.");
       break;
     default:
       System.out.println("I don't know which one to order.");
       break;
   }
 }
}
\par
class Main {
 public static void main(String[] args) {
   Test t1 = new Test(Size.MEDIUM);
   t1.orderPizza();
 }
}
```

```
I ordered a medium size pizza.
```

```java
enum Size{
   SMALL, MEDIUM, LARGE, EXTRALARGE;
\par
public String getSize() {
\par
// this will refer to the object SMALL
      switch(this) {
         case SMALL:
          return "small";
\par
case MEDIUM:
          return "medium";
\par
case LARGE:
          return "large";
\par
case EXTRALARGE:
          return "extra large";
\par
default:
          return null;
      }
   }
\par
public static void main(String[] args) {
\par
// calling the method getSize() using the object SMALL
      System.out.println("The size of the pizza is " + Size.SMALL.
    getSize());
   }
}
```

```java
class Size {
   public final static int SMALL = 1;
   public final static int MEDIUM = 2;
   public final static int LARGE = 3;
   public final static int EXTRALARGE = 4;
}
```

```
enum Size {
\par
// enum constants calling the enum constructors
   SMALL("The size is small."),
   MEDIUM("The size is medium."),
   LARGE("The size is large."),
   EXTRALARGE("The size is extra large.");
\par
private final String pizzaSize;
\par
// private enum constructor
   private Size(String pizzaSize) {
      this.pizzaSize = pizzaSize;
   }
\par
public String getSize() {
      return pizzaSize;
   }
}
\par
class Main {
   public static void main(String[] args) {
      Size size = Size.SMALL;
      System.out.println(size.getSize());
   }
}
```

```java
enum Size {
   SMALL, MEDIUM, LARGE, EXTRALARGE
}
\par
class Main {
   public static void main(String[] args) {
\par
System.out.println("string value of SMALL is " + Size.SMALL.toString
    ());
      System.out.println("string value of MEDIUM is " + Size.MEDIUM.
    name());
\par
}
}
```

```
string value of SMALL is SMALL
string value of MEDIUM is MEDIUM
```

```java
enum Size {
   SMALL {
\par
// overriding toString() for SMALL
      public String toString() {
        return "The size is small.";
      }
   },
\par
MEDIUM {
\par
// overriding toString() for MEDIUM
      public String toString() {
        return "The size is medium.";
      }
   };
}
\par
class Main {
   public static void main(String[] args) {
      System.out.println(Size.MEDIUM.toString());
   }
}
```