Module 3 – Core Java

1.Hello World

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

Sample Input: -

Sample Output: Hello, World!

2.Simple Calculator

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      int b = sc.nextInt();
      char ch = sc.next().charAt(0);
      switch(ch){
         case'+':
            System.out.println(a+b); break;
         case'-':
            System.out.println(a-b); break;
         case'*':
            System.out.println(a*b); break;
         case'/':
            System.out.println(a/b); break;
         case'%':
            System.out.println(a%b); break;
         default:
            break;
      }
   }
}
```

Sample Input: a = 5, b = 5, +

Sample Output: 10

3.Even or Odd

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int num = sc.nextInt();
      if(num % 2 == 0){
         System.out.println("The given number is even");
      }
      else{
         System.out.println("The given number is odd");
      }
   }
}
```

Sample Input: 999

Sample Output: The given number is odd

4.Leap Year

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int yr = sc.nextInt();
      if(yr%4 ==0 && yr %100 != 0 || yr %400 == 0){
         System.out.println("The given year is Leap Year");
      }
      else{
         System.out.println("The given year is not Leap Year");
      }
   }
}
```

Sample Input: 2004

Sample Output: The given year is Leap Year

5.Multiplication Table

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
```

```
    for(int i=1;i<=10;i++){
      System.out.println(a+" * "+i+" = "+i*a);
    }
  }
}
```

Sample Input: 5

Sample Output:

5 × 1 = 5
5 × 2 = 10
5 × 3 = 15
5 × 4 = 20
5 × 5 = 25
5 × 6 = 30
5 × 7 = 35
5 × 8 = 40
5 × 9 = 45
5 × 10 = 50

6.Data type demonstration

```
import java.util.*;
class Main{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    int num = 5;
    double dbt_num = 10;
    float fl_num = 5;
    char ch_ele = 'v';
    boolean flag = true;
    System.out.println(num);
    System.out.println(dbt_num);
    System.out.println(fl_num);
    System.out.println(ch_ele);
    System.out.println(flag);
  }
}
```

Sample Output:

5
10.0
5.0
v
true

7.Type Casting

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int num = 5;
      double dbt_num = 15.0;
      int double_int = (int)dbt_num;
      double int_double = num;
      System.out.println(double_int);
      System.out.println(int_double);
   }
}
```

Sample Input: -

Sample Output:

15

5.0

8.Operator Precedence

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int res = 10+5*2;
      System.out.println(res);

      //BODMAS(Brackets, Orders, Division, Multiplication, Addition, and Subtraction).
      // step 1 = 5*2 = 10
     // step 2 = 10 + 10 = 20
   }
}
```

Sample Output: 20

9.Grade Calculator

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int mrk = sc.nextInt();
      if(mrk >= 90) System.out.println("A");
      else if(mrk >= 80) System.out.println("B");
      else if(mrk >= 70) System.out.println("C");
```

```
      else if(mrk >= 60) System.out.println("D");
      else System.out.println("F");
   }
}
```

Sample Input: 46

Sample Output: F

10.Number Guessing game

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      Random r = new Random();
      int r_guess = r.nextInt(100)+1;
      int us_guess = 0;
      while(us_guess != r_guess){
         us_guess = sc.nextInt();
         if(us_guess > r_guess) System.out.println("Your guess is too high");
         else if(us_guess < r_guess)  System.out.println("Your guess is too low");
         else System.out.println("Congrats! You guessed right");
      }
   }
}
```

Sample Input1: 56

Sample Output1: Your guess is too low

Sample Input2: 59

Sample Output2: Your guess is too high

Sample Input3: 58

Sample Output3: Congrats! You guessed right

11.Factorail Calculator

```
import java.util.*;
class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int num = sc.nextInt();
      int fact = 1;
      for(int i = num;i>1;i--){
```

```
            fact = fact * i;
        }
        System.out.print(fact);
    }
}
```

Sample Input: 5

Sample Output: 120

12.Method Overloading

```
import java.util.*;
class Main{
    public static int add(int a,int b){
        return a+b;
    }
    public static int add(int a,int b,int c){
        return a+b+c;
    }
    public static double add(double d,double e){
        return d+e;
    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        int c = sc.nextInt();
        double d = sc.nextDouble();
        double e = sc.nextDouble();

        System.out.println(add(a,b));
        System.out.println(add(d,e));
        System.out.println(add(a,b,c));
    }
}
```

Sample Input:

a=5, b=6, c=2, d=10.0, e=15.0

Sample Output:

11

25.0

13

13.Recursive fibonacci

```
import java.util.*;
class Main{
   public static int fibonacci(int n){
      if(n<=1){
         return n;
      }
      return fibonacci(n-1)+fibonacci(n-2);
   }
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      for(int i =0;i<n;i++){
         System.out.println(fibonacci(i));
      }
   }
}
```

Sample Input: 5

Sample Output: 0 1 1 2 3

14.Array Sum and Average

```
import java.util.*;
class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] num = new int[n];
      System.out.println("Enter " + n + " elements:");
      for (int i = 0; i < n; i++) {
         num[i] = sc.nextInt();
      }
      int sum = 0;
      for (int number : num) {
         sum += number;
      }
      double average = (double) sum /n;
      System.out.println("Sum = " + sum);
      System.out.println("Average = " + average);
   }
}
```

Sample Input:

5

1 2 3 4 5

Sample Output:

Sum = 15

Average = 3

15.String Reversal

```
import java.util.*;
class Main{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String input = sc.nextLine();
      String reversed = new StringBuilder(input).reverse().toString();
      System.out.println("Reversed string: " + reversed);
   }
}
```

Sample Input: cognizant

Sample Output: tnazingoc

16.Palindrome Checker

```
import java.util.*;
class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String input = sc.nextLine();
      String cleaned = input.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();
      String reversed = new StringBuilder(cleaned).reverse().toString();
      if (cleaned.equals(reversed)) {
         System.out.println("The string is a palindrome.");
      } else {
         System.out.println("The string is not a palindrome.");
      }
   }
}
```

Sample Input1: Madam

Sample Output1: The string is a palindrome.

Sample Input2: Madam

Sample Output2: The string is not a palindrome.

17.class and object creation

```
import java.util.*;
public class Demo {
   static class Car {
      String make;
      String model;
      int year;
      public Car(String make, String model, int year) {
         this.make = make;
         this.model = model;
         this.year = year;
      }
      public void displayDetails() {
         System.out.println("Car Make: " + make);
         System.out.println("Car Model: " + model);
         System.out.println("Car Year: " + year);
         System.out.println();
      }
   }
   public static void main(String[] args) {
      Car car1 = new Car("Toyota", "Corolla", 2020);
      Car car2 = new Car("Ford", "Mustang", 2023);
      car1.displayDetails();
      car2.displayDetails();
   }
}
```

Sample Output:

Car Make: Toyota

Car Model: Corolla

Car Year: 2020

Car Make: Ford

Car Model: Mustang

Car Year: 2023

18. Inheritance

```
import java.util.*;
public class Inheritance {
   static class Animal {
      public void makeSound() {
         System.out.println("Some generic animal sound");
      }
   }
   static class Dog extends Animal {
      @Override
      public void makeSound() {
         System.out.println("Bark");
      }
   }
   public static void main(String[] args) {
      Animal genericAnimal = new Animal();
      Dog dog = new Dog();
      System.out.println("Animal says:");
      genericAnimal.makeSound();
      System.out.println("Dog says:");
      dog.makeSound();
   }
}
```

Sample Output:

```
Animal says: Some generic animal sound
Dog says: Bark
```

19.interface

```
import java.util.*;
public class InterfaceExample {
   interface Playable {
      void play();
   }
   static class Guitar implements Playable {
      public void play() {
         System.out.println("Strumming the guitar...");
      }
   }
   static class Piano implements Playable {
      public void play() {
         System.out.println("Playing the piano...");
      }
   }
   public static void main(String[] args) {
      Playable guitar = new Guitar();
```

```
        Playable piano = new Piano();
        System.out.println("Guitar:");
        guitar.play();
        System.out.println("Piano:");
        piano.play();
    }
}
```

Sample Output:

Guitar: Strumming the guitar…

Piano: Playing the piano...

20. Try-Catch

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            int num1 = sc.nextInt();
            int num2 = sc.nextInt();
            int result = num1 / num2;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Cannot divide by zero.");
        }
    }
}
```

Sample Input1: 10 5

Sample Output1: 2

Sample Input2: 2 0

Sample Output2: Error: Cannot divide by zero.

21.Custom Exception

```
import java.util.*;
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
```

```
}

public class CustomExceptionExample {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter your age: ");
      int age = sc.nextInt();

      try {
         if (age < 18) {
            throw new InvalidAgeException("Age must be at least 18.");
         }
         System.out.println("Age is valid.");
      } catch (InvalidAgeException e) {
         System.out.println("InvalidAgeException caught: " + e.getMessage());
      } finally {
         scanner.close();
      }
   }
}
```

Sample Input:

Enter your age: 20

Age is valid.

Sample Output:

Enter your age: 15

InvalidAgeException caught: Age must be at least 18.

22.File writing

```
import java.io.*;
import java.util.*;

public class FileWritingExample {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter a string to write to the file: ");
      String input = sc.nextLine();

      try (FileWriter writer = new FileWriter("output.txt")) {
         writer.write(input);
```

```
            System.out.println("Data has been written to output.txt");
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file.");
            e.printStackTrace();
        }
    }
}
```

Sample Input: Enter a string to write to the file: Hello, this is a test.

Sample Output:

Data has been written to **output.txt**

O

utput.txt: Hello, this is a test.

23.File Reading

```
import java.io.*;

public class FileReadingExample {
    public static void main(String[] args) {
        try (BufferedReader reader = new BufferedReader(new FileReader("output.txt"))) {
            String line;
            System.out.println("Contents of output.txt:");
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("An error occurred while reading the file.");
            e.printStackTrace();
        }
    }
}
```

Sample Input: Hello, this is a test.

Sample Output:

Contents of output.txt:

Hello, this is a test.

24.ArrayList

```
import java.util.*;
public class ArrayListExample {
   public static void main(String[] args) {
      ArrayList<String> studentNames = new ArrayList<>();
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter student names (type 'done' to finish):");
      while (true) {
         System.out.print("Enter name: ");
         String name = sc.nextLine();

         if (name.equalsIgnoreCase("done")) {
            break;
         }
         studentNames.add(name);
      }
      System.out.println("\nList of student names:");
      for (String student : studentNames) {
         System.out.println(student);
      }
   }
}
```

Sample Input:

Enter student names (type 'done' to finish):
Enter name: Alice
Enter name: Bob
Enter name: Charlie
Enter name: done

Sample Output:

List of student names:
Alice
Bob
Charlie

25.HashMap

```
import java.util.*;

public class HashMapExample {
   public static void main(String[] args) {
```

```
        HashMap<Integer, String> studentMap = new HashMap<>();
        Scanner sc = new Scanner(System.in);System.out.println("Add student entries (ID and
Name). Type -1 as ID to stop.");
        while (true) {
            System.out.print("Enter student ID (integer): ");
            int id = sc.nextInt();
            sc.nextLine();      // Consume newline

            if (id == -1) {
                break;
            }

            System.out.print("Enter student name: ");
            String name = sc.nextLine();

            studentMap.put(id, name);
        }
        System.out.print("\nEnter an ID to retrieve the student's name: ");
        int searchId = sc.nextInt();if (studentMap.containsKey(searchId)) {
            System.out.println("Student Name: " + studentMap.get(searchId));
        } else {
            System.out.println("No student found with ID " + searchId);
        }
    }
}
```

Sample Input:

Add student entries (ID and Name). Type -1 as ID to stop.

Enter student ID (integer): 101

Enter student name: Alice

Enter student ID (integer): 102

Enter student name: Bob

Enter student ID (integer): -1

Enter an ID to retrieve the student's name: 101

Sample Output:

Student Name: Alice

26.Thread

```
import java.lang.*;
class MessagePrinter implements Runnable {
   private String message;

   public MessagePrinter(String message) {
      this.message = message;
   }

   public void run() {
      for (int i = 0; i < 5; i++) {
         System.out.println(message + " - Count: " + (i + 1));
         try {
            Thread.sleep(500);
         } catch (InterruptedException e) {
            System.out.println("Thread interrupted");
         }
      }
   }
}
public class ThreadCreationExample {
   public static void main(String[] args) {
      Runnable task1 = new MessagePrinter("Thread 1 says Hello");
      Runnable task2 = new MessagePrinter("Thread 2 says Hi");
      Thread thread1 = new Thread(task1);
      Thread thread2 = new Thread(task2);
      thread1.start();
      thread2.start();
   }
}
```

Sample Output:

```
Thread 1 says Hello - Count: 1
Thread 2 says Hi - Count: 1
Thread 1 says Hello - Count: 2
Thread 2 says Hi - Count: 2
Thread 1 says Hello - Count: 3
Thread 2 says Hi - Count: 3
Thread 1 says Hello - Count: 4
Thread 2 says Hi - Count: 4
Thread 1 says Hello - Count: 5
Thread 2 says Hi - Count: 5
```

27.Lambda Experssions

```
import java.util.*;
public class LambdaSortExample {
   public static void main(String[] args) {
      List<String> names = new ArrayList<>();
      names.add("John");
      names.add("Alice");
      names.add("Bob");
      names.add("Diana");
      Collections.sort(names, (s1, s2) -> s1.compareToIgnoreCase(s2));
      System.out.println("Sorted list:");
      for (String name : names) {
         System.out.println(name);
      }
   }
}
```

Sample Output:

```
Sorted list:
Alice
Bob
Diana
John
```

28.Stream API

```
import java.util.*;
public class StreamFilterExample {
   public static void main(String[] args) {
      List<Integer> numbers = Arrays.asList(10, 15, 20, 25, 30, 35, 40);
      List<Integer> evenNumbers = numbers.stream()
                                 .filter(n -> n % 2 == 0)
                                 .collect(Collectors.toList());
      System.out.println("Even numbers: " + evenNumbers);
   }
}
```

Sample Output: Even numbers: [10, 20, 30, 40]

29.Records

```
import java.util.*;
public class RecordExample {
   public record Person(String name, int age) {}

   public static void main(String[] args) {
      Person p1 = new Person("Alice", 25);
      Person p2 = new Person("Bob", 17);
      Person p3 = new Person("Charlie", 30);
      System.out.println(p1);
      System.out.println(p2);
      System.out.println(p3);
      List<Person> people = List.of(p1, p2, p3);
      List<Person> adults = people.stream()
                           .filter(person -> person.age() >= 18)
                           .collect(Collectors.toList());

      System.out.println("\nAdults (age 18+):");
      adults.forEach(System.out::println);
   }
}
```

Sample Output:

```
Person[name=Alice, age=25]
Person[name=Bob, age=17]
Person[name=Charlie, age=30]

Adults (age 18+):
Person[name=Alice, age=25]
Person[name=Charlie, age=30]
```

30.Pattern

```
 public class PatternMatchingSwitchExample {

   public static void checkObjectType(Object obj) {
      String result = switch (obj) {
         case Integer i -> "Integer with value: " + i;
         case String s -> "String with content: \"" + s + "\"";
         case Double d -> "Double with value: " + d;
         case null -> "Object is null";
         default -> "Unknown type: " + obj.getClass().getSimpleName();
      };
      System.out.println(result);
   }
```

```
    public static void main(String[] args) {
        checkObjectType(42);
        checkObjectType("Hello");
        checkObjectType(3.14);
        checkObjectType(true);
        checkObjectType(null);
    }
}
```

Sample Output:

```
Integer with value: 42
String with content: "Hello"
Double with value: 3.14
Unknown type: Boolean
Object is null
```

31. Basic JDBC Connection

```
import java.sql.*;

public class JdbcExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/school";
        String user = "root";
        String password = "your_password";

        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection(url, user, password);
            stmt = conn.createStatement();
            String sql = "SELECT * FROM students";
            rs = stmt.executeQuery(sql);
            System.out.println("ID\tName\tAge");
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                int age = rs.getInt("age");

                System.out.printf("%d\t%s\t%d%n", id, name, age);
            }
```

```
        } catch (ClassNotFoundException e) {
            System.out.println("MySQL JDBC Driver not found. Include it in your library path.");
            e.printStackTrace();
        } catch (SQLException e) {
            System.out.println("Database error:");
            e.printStackTrace();
        } finally {
            try { if (rs != null) rs.close(); } catch (SQLException ignored) {}
            try { if (stmt != null) stmt.close(); } catch (SQLException ignored) {}
            try { if (conn != null) conn.close(); } catch (SQLException ignored) {}
        }
    }
}
```

Sample Output:

ID	Name	Age
1	Alice	20
2	Bob	22
3	Charlie	19

32.Insert and Update operations in JDBC

```
import java.sql.*;

public class StudentDAOExample {
    private static final String URL = "jdbc:mysql://localhost:3306/school";
    private static final String USER = "root";
    private static final String PASSWORD = "your_password";
    public boolean insertStudent(int id, String name, int age) {
        String sql = "INSERT INTO students (id, name, age) VALUES (?, ?, ?)";
        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, id);
            pstmt.setString(2, name);
            pstmt.setInt(3, age);
            int rowsInserted = pstmt.executeUpdate();
            return rowsInserted > 0;
        } catch (SQLException e) {
            System.err.println("Insert failed: " + e.getMessage());
            return false;
        }
    }
```

```
    public boolean updateStudent(int id, String newName, int newAge) {
        String sql = "UPDATE students SET name = ?, age = ? WHERE id = ?";
        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, newName);
            pstmt.setInt(2, newAge);
            pstmt.setInt(3, id);
            int rowsUpdated = pstmt.executeUpdate();
            return rowsUpdated > 0;
        } catch (SQLException e) {
            System.err.println("Update failed: " + e.getMessage());
            return false;
        }
    }

    public static void main(String[] args) {
        StudentDAOExample dao = new StudentDAOExample();
        if (dao.insertStudent(4, "David", 21)) {
            System.out.println("Student inserted successfully.");
        } else {
            System.out.println("Failed to insert student.");
        }
        if (dao.updateStudent(4, "David Smith", 22)) {
            System.out.println("Student updated successfully.");
        } else {
            System.out.println("Failed to update student.");
        }
    }
}
```

Sample Output:

Student inserted successfully.

Student updated successfully.

33.Transaction Handling in jdbc

```
import java.sql.*;

public class TransactionExample {
    private static final String URL = "jdbc:mysql://localhost:3306/bank";
    private static final String USER = "root";
    private static final String PASSWORD = "your_password";

    public static void transferMoney(int fromAccount, int toAccount, double amount) {
```

```
    String debitSql = "UPDATE accounts SET balance = balance - ? WHERE account_id = ?
AND balance >= ?";
    String creditSql = "UPDATE accounts SET balance = balance + ? WHERE account_id = ?";

    try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD)) {
      conn.setAutoCommit(false);

      try (PreparedStatement debitStmt = conn.prepareStatement(debitSql);
           PreparedStatement creditStmt = conn.prepareStatement(creditSql)) {
        debitStmt.setDouble(1, amount);
        debitStmt.setInt(2, fromAccount);
        debitStmt.setDouble(3, amount);
        int debitResult = debitStmt.executeUpdate();

        if (debitResult != 1) {
          throw new SQLException("Insufficient funds or account not found for debit.");
        }
        creditStmt.setDouble(1, amount);
        creditStmt.setInt(2, toAccount);
        int creditResult = creditStmt.executeUpdate();

        if (creditResult != 1) {
          throw new SQLException("Receiver account not found for credit.");
        }

        conn.commit();
        System.out.println("Transfer successful.");

      } catch (SQLException e) {
        conn.rollback();
        System.err.println("Transfer failed: " + e.getMessage());
      } finally {
        conn.setAutoCommit(true);
      }

    } catch (SQLException e) {
      System.err.println("Database connection error: " + e.getMessage());
    }
  }

  public static void main(String[] args) {
    transferMoney(1, 2, 200.0);
  }
}
```

Sample Output1:

Transfer successful.

Sample Output2:

Transfer failed: Insufficient funds or account not found for debit.

34.Create and use java modules

```
public class ModuleSimulation {

   // Simulating com.utils.StringUtils

   static class StringUtils {
      public static String shout(String message) {
         return message.toUpperCase() + "!!!";
      }
   }

   // Simulating com.greetings.Main

   public static void main(String[] args) {
      String greeting = "hello from modules";
      System.out.println(StringUtils.shout(greeting));
   }
}
```

Sample Output: HELLO FROM MODULES!!!

35.TCP Client-Server chat

(i)server side

```
import java.io.*;
import java.net.*;
public class ChatServer {
   public static void main(String[] args) {
      try (ServerSocket serverSocket = new ServerSocket(5000)) {
         System.out.println("Server started. Waiting for client...");
         Socket socket = serverSocket.accept();
         System.out.println("Client connected.");

         BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
         PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);
         BufferedReader consoleInput = new BufferedReader(new
InputStreamReader(System.in));
```

```
        // Thread to read messages from client
        new Thread(() -> {
            String msgFromClient;
            try {
                while ((msgFromClient = reader.readLine()) != null) {
                    System.out.println("Client: " + msgFromClient);
                }
            } catch (IOException e) {
                System.out.println("Client disconnected.");
            }
        }).start();

        // Main thread to send messages
        String msgToClient;
        while ((msgToClient = consoleInput.readLine()) != null) {
            writer.println(msgToClient);
        }

    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

Server console:

```
Server started. Waiting for client...
Client connected.
Client: Hello Server!
Hi Client, how are you?
Client: I'm good, thanks!
Good to hear that.
```

(ii)client side

```
import java.io.*;
import java.net.*;

public class ChatClient {
  public static void main(String[] args) {
    try (Socket socket = new Socket("localhost", 5000)) {
        System.out.println("Connected to server.");

        BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);
        BufferedReader consoleInput = new BufferedReader(new
InputStreamReader(System.in));
```

```
        // Thread to read messages from server
        new Thread(() -> {
            String msgFromServer;
            try {
                while ((msgFromServer = reader.readLine()) != null) {
                    System.out.println("Server: " + msgFromServer);
                }
            } catch (IOException e) {
                System.out.println("Server disconnected.");
            }
        }).start();

        // Main thread to send messages
        String msgToServer;
        while ((msgToServer = consoleInput.readLine()) != null) {
            writer.println(msgToServer);
        }

    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

Client console:

```
Connected to server.
Hello Server!
Server: Hi Client, how are you?
I'm good, thanks!
Server: Good to hear that.
```

36.HTTP Client API

```
import java.io.*;
import java.net.*;
import com.google.gson.*

public class GitHubHttpClient {
  public static void main(String[] args) {
    // Create HTTP client
    HttpClient client = HttpClient.newHttpClient();
    // Define HTTP request
    HttpRequest request = HttpRequest.newBuilder()
        .uri(URI.create("https://api.github.com/repos/openai/gpt-3"))
        .header("Accept", "application/vnd.github.v3+json")
        .build();
```

```
        try {
            // Send request and get response
            HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());

            // Print status code and raw body
            System.out.println("Status Code: " + response.statusCode());
            System.out.println("Response Body:\n" + response.body());

            // Parse JSON (optional)
            JsonObject json = JsonParser.parseString(response.body()).getAsJsonObject();
            System.out.println("\nRepository Name: " + json.get("name").getAsString());
            System.out.println("Stars: " + json.get("stargazers_count").getAsInt());
            System.out.println("Description: " + json.get("description").getAsString());

        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Sample Output:

```
Status Code: 200
Response Body:
{
  ... full JSON response ...
}


Repository Name: gpt-3
Stars: 10000
Description: GPT-3 is OpenAI's third-generation language prediction model.
```

37.Using javap to inspect Bytecode

```
import java.io.*;
import java.nio.file.*;
public class JavapDemo {
    public static void main(String[] args) throws Exception {

        // Step 1: Write a simple Java class to file
        String className = "TempClass";
        String javaCode = """
            public class TempClass {
                public static void main(String[] args) {
                    System.out.println("Hello from bytecode!");
```

```
        }
    }
    """;
    Files.writeString(Path.of(className + ".java"), javaCode);

    // Step 2: Compile the Java file
    Process compile = new ProcessBuilder("javac", className + ".java")
            .inheritIO().start();
    compile.waitFor();

    // Step 3: Run javap -c to see bytecode
    System.out.println("\nBytecode for " + className + ":");
    Process javap = new ProcessBuilder("javap", "-c", className)
            .redirectErrorStream(true)
            .start();

    // Step 4: Display the output
    try (BufferedReader reader = new BufferedReader(
            new InputStreamReader(javap.getInputStream()))) {
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
    }

    // Optional: Clean up files (uncomment if you want cleanup)
    // Files.deleteIfExists(Path.of(className + ".java"));
    // Files.deleteIfExists(Path.of(className + ".class"));
  }
}
```

Sample Output:

```
Bytecode for TempClass:
Compiled from "TempClass.java"
public class TempClass {
  public TempClass();
    Code:
      0: aload_0
      1: invokespecial #1                  // Method java/lang/Object."<init>":()V
      4: return

  public static void main(java.lang.String[]);
    Code:
      0: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
      3: ldc           #3                  // String Hello from bytecode!
```

```
    5: invokevirtual #4                  // Method java/io/PrintStream.println:(Ljava/lang/String;)V
    8: return
}
```

38.Decompose a class Files

```
import java.io.*;
import java.nio.file.*;

public class DecompilerDemo {
   public static void main(String[] args) throws Exception {
      String javaSource = """
         public class Temp {
            public static void main(String[] args) {
               System.out.println("Hello from compiled class!");
            }
         }
         """;

      // Write Temp.java
      Files.writeString(Path.of("Temp.java"), javaSource);

      // Compile Temp.java
      Process compile = new ProcessBuilder("javac", "Temp.java")
            .inheritIO().start();
      compile.waitFor();

     // Run CFR to decompile Temp.class
      System.out.println("\n=== Decompiled Output via CFR ===");
      Process decompile = new ProcessBuilder(
            "java", "-jar", "cfr.jar", "Temp.class", "--silent", "true"
      ).redirectErrorStream(true).start();

      // Print decompiled output
      try (BufferedReader reader = new BufferedReader(
            new InputStreamReader(decompile.getInputStream()))) {
         String line;
         while ((line = reader.readLine()) != null) {
            System.out.println(line);
         }
      }
   }
}
```

Sample Output:

```
// Decompiled by CFR 0_151
```

```
public class Temp {
    public static void main(String[] args) {
        System.out.println("Hello from compiled class!");
    }
}
```

39.Reflection in java

```
import java.lang.reflect.Method;

public class SimpleReflection {
    public static void main(String[] args) throws Exception {
        // Load MyClass
        Class<?> clazz = Class.forName("MyClass");

        // Create an instance of MyClass
        Object obj = clazz.getDeclaredConstructor().newInstance();

        // Get all declared methods
        for (Method method : clazz.getDeclaredMethods()) {
            System.out.println("Found method: " + method.getName());

            // Invoke method 'sayHello' if found
            if (method.getName().equals("sayHello")) {
                method.invoke(obj); // call method with no arguments
            }
        }
    }
}

// Class to reflect on
class MyClass {
    public void sayHello() {
        System.out.println("Hello from reflection!");
    }
}
```

Sample Output:

```
Found method: sayHello
Hello from reflection!
```

40.virtual threads

```
public class SimpleVirtualThreads {
    public static void main(String[] args) throws InterruptedException {
        Thread[] threads = new Thread[10];
```

```
        for (int i = 0; i < 10; i++) {
            threads[i] = Thread.startVirtualThread(() -> {
                System.out.println("Hello from virtual thread: " +
Thread.currentThread().threadId());
            });
        }

        for (Thread t : threads) {
            t.join(); // wait for all threads to finish
        }
    }
}
```

Sample Output:

```
Hello from virtual thread: 12
Hello from virtual thread: 13
Hello from virtual thread: 14
Hello from virtual thread: 15
Hello from virtual thread: 16
Hello from virtual thread: 17
Hello from virtual thread: 18
Hello from virtual thread: 19
Hello from virtual thread: 20
Hello from virtual thread: 21
```

41.Executor Service and Callable

```
import java.util.concurrent.*;
import java.util.*;

public class CallableExecutorExample {
    public static void main(String[] args) throws Exception {
        // Create a fixed thread pool with 3 threads
        ExecutorService executor = Executors.newFixedThreadPool(3);
        // List to hold Future results
        List<Future<String>> futures = new ArrayList<>();
        // Submit 5 Callable tasks
        for (int i = 1; i <= 5; i++) {
            int taskId = i;
            Callable<String> task = () -> {
                // Simulate some work
```

```
            Thread.sleep(500);
            return "Result from task " + taskId;
        };
        futures.add(executor.submit(task));
    }
    // Retrieve and print results
    for (Future<String> future : futures) {
        System.out.println(future.get()); // blocks until result available
    }
    // Shutdown executor
    executor.shutdown();
  }
}
```

Sample Output:

Result from task 1
Result from task 2
Result from task 3
Result from task 4
Result from task 5