JAVA 5

1. Write a program to copy the contents of one file to another using FileInputStream and FileOutputStream.

```
import java.io.*;

public class FileCopy {
    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("source.txt");
        FileOutputStream fos = new FileOutputStream("destination.txt");

        int byteData;
        while ((byteData = fis.read()) != -1) {
            fos.write(byteData);
        }

        fis.close();
        fos.close();
    }
}
```

---

2. Write a program to read and write character data using FileReader and FileWriter.

```
import java.io.*;

public class FileReaderWriter {
    public static void main(String[] args) throws IOException {
        FileReader fr = new FileReader("source.txt");
        FileWriter fw = new FileWriter("destination.txt");

        int charData;
        while ((charData = fr.read()) != -1) {
            fw.write(charData);
        }

        fr.close();
        fw.close();
    }
}
```

---

3. Write a program to read a text file line by line using BufferedReader and write to another file using BufferedWriter.

```
import java.io.*;

public class LineByLineFileCopy {
```

```
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader("source.txt"));
        BufferedWriter bw = new BufferedWriter(new FileWriter("destination.txt"));

        String line;
        while ((line = br.readLine()) != null) {
            bw.write(line);
            bw.newLine();
        }

        br.close();
        bw.close();
    }
}
```

---

4. Write a program to store and retrieve student details (name, age, grade) using DataOutputStream and DataInputStream.

```
import java.io.*;

public class StudentDetails {
    public static void main(String[] args) throws IOException {
        DataOutputStream dos = new DataOutputStream(new
FileOutputStream("student.dat"));
        dos.writeUTF("John");
```

```
        dos.writeInt(20);
        dos.writeUTF("A");
        dos.close();

        DataInputStream dis = new DataInputStream(new FileInputStream("student.dat"));
        System.out.println("Name: " + dis.readUTF());
        System.out.println("Age: " + dis.readInt());
        System.out.println("Grade: " + dis.readUTF());
        dis.close();
    }
}
```

output : Name: John

Age: 20

Grade: A

---

5. Write a program to serialize and deserialize a Book object using ObjectOutputStream and ObjectInputStream.

```
import java.io.*;

class Book implements Serializable {
    String title;
    String author;
```

```
    Book(String title, String author) {
        this.title = title;
        this.author = author;
    }
}

public class SerializeBook {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        // Serialize
        Book book = new Book("Java Programming", "John Doe");
        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("book.ser"));
        oos.writeObject(book);
        oos.close();

        // Deserialize
        ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("book.ser"));
        Book deserializedBook = (Book) ois.readObject();
        System.out.println("Title: " + deserializedBook.title);
        System.out.println("Author: " + deserializedBook.author);
        ois.close();
    }
}
```

output : Title: Java Programming

Author: John Doe

---

6. Write a program to read a file and handle FileNotFoundException, IOException, and a custom exception for empty files.

```
import java.io.*;

class EmptyFileException extends Exception {
    public EmptyFileException(String message) {
        super(message);
    }
}

public class FileReaderException {
    public static void main(String[] args) {
        try {
            FileReader fr = new FileReader("source.txt");
            int data = fr.read();
            if (data == -1) {
                throw new EmptyFileException("File is empty.");
            }
            fr.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IOException occurred: " + e.getMessage());
        } catch (EmptyFileException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
}
```

---

7. Write a program to copy a text file using BufferedReader & BufferedWriter and a binary file using FileInputStream & FileOutputStream.

```
import java.io.*;

public class FileCopy {
    public static void main(String[] args) throws IOException {
        // Text File Copy
        BufferedReader br = new BufferedReader(new FileReader("source.txt"));
        BufferedWriter bw = new BufferedWriter(new FileWriter("destination.txt"));
        String line;
        while ((line = br.readLine()) != null) {
            bw.write(line);
            bw.newLine();
        }
        br.close();
        bw.close();

        // Binary File Copy
```

```
        FileInputStream fis = new FileInputStream("source.bin");
        FileOutputStream fos = new FileOutputStream("destination.bin");
        int byteData;
        while ((byteData = fis.read()) != -1) {
            fos.write(byteData);
        }
        fis.close();
        fos.close();
    }
}
```

---

8. Write a program to read a large CSV file line by line using BufferedReader and process the data efficiently.

```
import java.io.*;

public class CSVReader {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader("data.csv"));
        String line;
        while ((line = br.readLine()) != null) {
            String[] data = line.split(",");
            System.out.println("Name: " + data[0] + ", Age: " + data[1]);
        }
```

```
        br.close();
    }
}
```

---

9. Write a program to implement a simple text editor where users can create, edit, and read files using Scanner, FileWriter, and BufferedReader.

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

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

        // Create or edit file
        System.out.print("Enter text to write into file: ");
        String text = sc.nextLine();
        FileWriter fw = new FileWriter("editor.txt");
        fw.write(text);
        fw.close();

        // Read file
        BufferedReader br = new BufferedReader(new FileReader("editor.txt"));
        String fileContent = br.readLine();
```

```
        System.out.println("File Content: " + fileContent);
        br.close();
    }
}
```

---

10. Write a program to implement a user authentication system using object serialization (ObjectOutputStream, ObjectInputStream).

```
import java.io.*;

class User implements Serializable {
    String username;
    String password;

    User(String username, String password) {
        this.username = username;
        this.password = password;
    }
}

public class UserAuthentication {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        // Serialize User
        User user = new User("john", "password123");
```

```
        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("user.ser"));
        oos.writeObject(user);
        oos.close();

        // Deserialize User
        ObjectInputStream ois = new ObjectInputStream(new FileInputStream("user.ser"));
        User deserializedUser = (User) ois.readObject();
        System.out.println("Username: " + deserializedUser.username);
        System.out.println("Password: " + deserializedUser.password);
        ois.close();
    }
}
```

---

11. Write a program to demonstrate ArithmeticException by performing division by zero and handling it using try-catch.

```
public class ArithmeticExceptionExample {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero");
        }
    }
```

```
}
```

output: Error: Division by zero

---

12. Write a program to handle ArrayIndexOutOfBoundsException by accessing an invalid index in an array.

```
public class ArrayIndexOutOfBoundsExample {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3};
        try {
            System.out.println(arr[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Invalid array index");
        }
    }
}
```

output: Error: Invalid array index

---

13. Write a program to demonstrate multiple catch blocks by handling NullPointerException and NumberFormatException.

```
public class MultipleCatchExample {
    public static void main(String[] args) {
        try {
            String str = null;
            System.out.println(str.length());
            int num = Integer.parseInt("abc");
        } catch (NullPointerException e) {
            System.out.println("Error: Null pointer exception");
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid number format");
        }
    }
}
```

output: Error: Null pointer exception

---

14. Write a program to demonstrate nested try-catch blocks, handling exceptions at different levels.

```
public class NestedTryCatch {
    public static void main(String[] args) {
        try {
            try {
                int result = 10 / 0;
            } catch (ArithmeticException e) {
```

```
            System.out.println("Inner catch: Division by zero");
        }
    } catch (Exception e) {
        System.out.println("Outer catch: Exception occurred");
    }
  }
}
```

output: Inner catch: Division by zero

---

15. Write a program to illustrate throw and throws by creating a method that throws IllegalArgumentException if input is negative.

```
public class ThrowAndThrows {
  public static void main(String[] args) {
    try {
      checkAge(-5);
    } catch (IllegalArgumentException e) {
      System.out.println(e.getMessage());
    }
  }

  public static void checkAge(int age) throws IllegalArgumentException {
    if (age < 0) {
      throw new IllegalArgumentException("Age cannot be negative");
```

```
        }
    }
}
```

output: Age cannot be negative

---

16. Write a program to use the finally block to ensure resource closure after a file read operation.

```
import java.io.*;

public class FinallyBlockExample {
    public static void main(String[] args) {
        BufferedReader br = null;
        try {
            br = new BufferedReader(new FileReader("test.txt"));
            String line = br.readLine();
            System.out.println(line);
        } catch (IOException e) {
            System.out.println("IOException occurred");
        } finally {
            try {
                if (br != null) {
                    br.close();
                }
```

```
            } catch (IOException e) {
                System.out.println("Error closing the file");
            }
        }
    }
}
```

---

17. Write a program to demonstrate custom exception handling by creating a user-defined exception class InvalidAgeException for voting eligibility.

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}

public class VotingEligibility {
    public static void main(String[] args) {
        try {
            checkEligibility(15);
        } catch (InvalidAgeException e) {
            System.out.println(e.getMessage());
        }
    }
```

```
    public static void checkEligibility(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("You must be at least 18 years old to vote.");
        } else {
            System.out.println("You are eligible to vote.");
        }
    }
}
```

output: You must be at least 18 years old to vote.

---

18. Write a program to demonstrate exception propagation using multiple method calls.

```
public class ExceptionPropagation {
    public static void main(String[] args) {
        try {
            methodA();
        } catch (Exception e) {
            System.out.println("Exception handled in main: " + e.getMessage());
        }
    }

    public static void methodA() throws Exception {
        methodB();
```

```
    }

    public static void methodB() throws Exception {
        throw new Exception("Exception occurred in methodB");
    }
}
```

output: Exception handled in main: Exception occurred in methodB

---

19. Write a program to simulate an online banking system that throws InsufficientFundsException if withdrawal amount exceeds balance.

```
class InsufficientFundsException extends Exception {
    public InsufficientFundsException(String message) {
        super(message);
    }
}

public class BankingSystem {
    public static void main(String[] args) {
        try {
            withdraw(5000, 3000);
        } catch (InsufficientFundsException e) {
            System.out.println(e.getMessage());
        }
```

```
    }

    public static void withdraw(int balance, int amount) throws InsufficientFundsException
{
        if (amount > balance) {
            throw new InsufficientFundsException("Insufficient funds for the withdrawal");
        } else {
            System.out.println("Withdrawal successful");
        }
    }
}
```

output: Withdrawal successful

---

20. Write a program to handle Java's built-in exceptions like IOException while reading a file.

```
import java.io.*;

public class IOExceptionExample {
    public static void main(String[] args) {
        try {
            FileReader fr = new FileReader("nonexistent.txt");
            fr.read();
        } catch (IOException e) {
            System.out.println("IOException occurred: " + e.getMessage());
```

```
        }
    }
}
```

output:
