1. Write a program to create a new text file named test.txt.

```
import java.io.File;
import java.io.IOException;
public class CreateFile {
   public static void main(String[] args) throws IOException {
      File f = new File("test.txt");
      if (f.createNewFile()) {
         System.out.println("File created: " + f.getName());
      } else {
         System.out.println("File already exists.");
      }
   }
}
```

Output:

```
File created: test.txt
```

2. Write a program to check whether a file exists at a given path.

```
import java.io.File;
public class CheckFileExists {
   public static void main(String[] args) {
      File f = new File("test.txt");
      if (f.exists()) {
         System.out.println("File exists.");
      } else {
         System.out.println("File does not exist.");
      }
   }
}
```

Output:

```
File exists.
```

3. Write a Java program to write "Hello, World!" into a file using FileWriter.

```
import java.io.FileWriter;
```

```
import java.io.IOException;


public class WriteFile {
   public static void main(String[] args) {
      try {
         FileWriter fw = new FileWriter("test.txt");
         fw.write("Hello, World!");
         fw.close();
         System.out.println("Written 'Hello, World!' to file.");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

Output:

```
Written 'Hello, World!' to file.
```

4. Write a program to read the content of a file line by line using BufferedReader.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;


public class ReadFile {
   public static void main(String[] args) throws IOException {
      BufferedReader br = new BufferedReader(new FileReader("test.txt"));
      String line;
      while ((line = br.readLine()) != null) {
         System.out.println("Read line: " + line);
      }
      br.close();
   }
}
```

Output:

Read line: Hello, World!

5. Write a program to append a line of text to an existing file.

```
import java.io.FileWriter;
import java.io.IOException;
public class AppendFile {
   public static void main(String[] args) {
      try {
         FileWriter fw = new FileWriter("test.txt", true);
         fw.write("\nThis is appended line.");
         fw.close();
         System.out.println("Appended line to file.");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

Output:

Appended line to file.

6. Write a program to count the number of lines, words, and characters in a file.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class CountFile {
   public static void main(String[] args) throws IOException {
      BufferedReader br = new BufferedReader(new FileReader("test.txt"));
      int lines = 0, words = 0, chars = 0;
      String line;
      while ((line = br.readLine()) != null) {
         lines++;
```

```
            String[] w = line.split("\\s+");
            words += w.length;
            chars += line.length();
        }
        br.close();
        System.out.println("Lines: " + lines + ", Words: " + words + ", Characters: " + chars);
    }
}
```

Output:

Lines: 2, Words: 6, Characters: 38

7. Write a program to copy content from one file to another using FileReader and FileWriter.

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class CopyFile {
    public static void main(String[] args) throws IOException {
        FileReader fr = new FileReader("test.txt");
        FileWriter fw = new FileWriter("copy.txt");
        int c;
        while ((c = fr.read()) != -1) {
            fw.write(c);
        }
        fr.close();
        fw.close();
        System.out.println("File copied to copy.txt");
    }
}
```

Output:

File copied to copy.txt

8. Write a program that lists all the files in a directory.

```
import java.io.File;
public class ListFiles {
   public static void main(String[] args) {
      File dir = new File(".");
      String[] files = dir.list();
      System.out.println("Files in current directory:");
      for (String f : files) {
         System.out.println(f);
      }
   }
}
```

Output: (example output depends on your directory)

```
Files in current directory:
test.txt
copy.txt
student.ser
CreateFile.class
...
```

9. Write a program to filter and display only .txt files from a folder using FilenameFilter.

```
import java.io.File;
import java.io.FilenameFilter;
public class FilterTxtFiles {
   public static void main(String[] args) {
      File dir = new File(".");
      FilenameFilter filter = new FilenameFilter() {
         public boolean accept(File dir, String name) {
            return name.endsWith(".txt");
         }
      };
      String[] txtFiles = dir.list(filter);
```

```
        System.out.println(".txt files:");
        for (String f : txtFiles) {
            System.out.println(f);
        }
    }
}
```

Output:

.txt files:

test.txt

copy.txt

10. Write a program to serialize and deserialize a Student object to and from a file.

```
import java.io.*;
class Student implements Serializable {
    int id;
    String name;

    Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public String toString() {
        return id + " " + name;
    }
}

public class SerializeStudent {
    public static void main(String[] args) throws Exception {
        Student s = new Student(1, "Alice");

        // Serialize
```

```
        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("student.ser"));
        oos.writeObject(s);
        oos.close();
        System.out.println("Serialized Student object.");

        // Deserialize
        ObjectInputStream ois = new ObjectInputStream(new FileInputStream("student.ser"));
        Student s1 = (Student) ois.readObject();
        ois.close();
        System.out.println("Deserialized Student object: " + s1);
    }
}
```

Output:

Serialized Student object.

Deserialized Student object: 1 Alice

11. Write a program to read a file using Scanner and display the tokens.

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

public class ReadFileUsingScanner {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(new File("test.txt"));
        System.out.println("Tokens in test.txt:");
        while (sc.hasNext()) {
            System.out.println(sc.next());
        }
        sc.close();
    }
}
```

Output:

Tokens in test.txt:

Hello,

World!

This

is

appended

line.

12. Write a program to search for a specific word in a file and count its occurrences.

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


public class SearchWordInFile {
   public static void main(String[] args) throws Exception {
      String wordToFind = "Hello";
      Scanner sc = new Scanner(new File("test.txt"));
      int count = 0;
      while (sc.hasNext()) {
         if (sc.next().equals(wordToFind)) {
            count++;
         }
      }
      sc.close();
      System.out.println("Word '" + wordToFind + "' found " + count + " times.");
   }
}
```

Output:

Word 'Hello' found 1 times.

13. Write a program to create, move, and delete a file using Files and Paths.

```
import java.nio.file.*;
public class FileOperations {
   public static void main(String[] args) throws Exception {
      Path path = Paths.get("fileToMove.txt");
```

```
        Files.createFile(path);
        System.out.println("Created file fileToMove.txt");

        Path movedPath = Paths.get("movedFile.txt");
        Files.move(path, movedPath, StandardCopyOption.REPLACE_EXISTING);
        System.out.println("Moved fileToMove.txt to movedFile.txt");

        Files.deleteIfExists(movedPath);
        System.out.println("Deleted movedFile.txt");
    }
}
```

Output:

```
Created file fileToMove.txt
Moved fileToMove.txt to movedFile.txt
Deleted movedFile.txt
```

14. Write a program to read all lines of a file using Files.readAllLines() and print them.

```
import java.nio.file.*;
import java.util.List;
public class ReadAllLines {
    public static void main(String[] args) throws Exception {
        List<String> lines = Files.readAllLines(Paths.get("test.txt"));
        System.out.println("All lines from test.txt:");
        for (String l : lines) {
            System.out.println(l);
        }
    }
}
```

Output:

```
All lines from test.txt:
Hello, World!
This is appended line.
```

Appended using Files.write

15. Write a program to write data into a file using Files.write() and append using StandardOpenOption.APPEND.

```
import java.nio.file.*;
import java.nio.file.StandardOpenOption;
public class WriteAppendFile {
   public static void main(String[] args) throws Exception {
      String data = "\nAppended using Files.write";
      Files.write(Paths.get("test.txt"), data.getBytes(), StandardOpenOption.APPEND);
      System.out.println("Appended line using Files.write");
   }
}
```

Output:

Appended line using Files.write

16. Write a program to walk through a directory tree and display file names using Files.walk().

```
import java.nio.file.*;
public class WalkDirectory {
   public static void main(String[] args) throws Exception {
      System.out.println("Files in directory tree:");
      Files.walk(Paths.get("."))
          .filter(Files::isRegularFile)
          .forEach(System.out::println);
   }
}
```

Output:

Files in directory tree:

.\test.txt

.\copy.txt

.\student.ser

.\employee.ser

.\testCopy.txt

...

17. Write a program to copy a file using Files.copy() with REPLACE_EXISTING option.

```
import java.nio.file.*;
public class CopyFileNIO {
   public static void main(String[] args) throws Exception {
      Files.copy(Paths.get("test.txt"), Paths.get("testCopy.txt"),
StandardCopyOption.REPLACE_EXISTING);
      System.out.println("Copied test.txt to testCopy.txt");
   }
}
```

Output:

```
Copied test.txt to testCopy.txt
```

18. Write a program to check and print the size of a file in bytes using Files.size().

```
import java.nio.file.*;
public class FileSize {
   public static void main(String[] args) throws Exception {
      long size = Files.size(Paths.get("test.txt"));
      System.out.println("Size of test.txt: " + size + " bytes");
   }
}
```

Output:

```
Size of test.txt: 79 bytes
```

19. Write a program to serialize a class Employee and store it in employee.ser.

```
import java.io.*;
class Employee implements Serializable {
   int id;
   String name;

   Employee(int id, String name) {
      this.id = id;
      this.name = name;
```

```
    }

    public String toString() {
        return id + " " + name;
    }
}

public class SerializeEmployee {
    public static void main(String[] args) throws Exception {
        Employee emp = new Employee(101, "Bob");

        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("employee.ser"));
        oos.writeObject(emp);
        oos.close();
        System.out.println("Serialized Employee object.");
    }
}
```

Output:

```
Serialized Employee object.
```

20. Write a program to deserialize the employee.ser file and display the object data.

```
import java.io.*;
public class DeserializeEmployee {
    public static void main(String[] args) throws Exception {
        ObjectInputStream ois = new ObjectInputStream(new FileInputStream("employee.ser"));
        Employee emp = (Employee) ois.readObject();
        ois.close();
        System.out.println("Deserialized Employee: " + emp);
    }
}
```

Output:

```
Deserialized Employee: 101 Bob
```