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

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

public class CreateFile {
  public static void main(String[] args) {
    try {
      FileWriter writer = new FileWriter("test.txt");
      writer.write("This is a new text file named test.txt.");
      writer.close();
      System.out.println("File 'test.txt' has been created.");
    } catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

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

```
import java.io.File;

public class FileExistsCheck {
  public static void main(String[] args) {
    String path = "test.txt";
    File file = new File(path);

    if (file.exists()) {
      System.out.println("File exists at the given path.");
    } else {
      System.out.println("File does not exist at the given path.");
    }
  }
}
```

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

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

public class HelloWorldToFile {
   public static void main(String[] args) {
      try {
         FileWriter writer = new FileWriter("hello.txt");
         writer.write("Hello, World!");
         writer.close();
         System.out.println("Successfully wrote to the file.");
      } catch (IOException e) {
         System.out.println("An error occurred.");
         e.printStackTrace();
      }
   }
}
```

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 ReadFileLineByLine {
   public static void main(String[] args) {
      String path = "hello.txt";

      try {
         BufferedReader reader = new BufferedReader(new FileReader(path));
         String line;

         while ((line = reader.readLine()) != null) {
            System.out.println(line);
 }

         reader.close();
      } catch (IOException e) {
         System.out.println("An error occurred while reading the file.");
         e.printStackTrace();
      }
   }
}
```

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

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

```
import java.io.IOException;

public class AppendToFile {
  public static void main(String[] args) {
    String path = "hello.txt";
    String textToAppend = "This is an appended line.";

    try {
      FileWriter writer = new FileWriter(path, true);
      writer.write("\n" + textToAppend);
      writer.close();
      System.out.println("Text appended to the file successfully.");
    } catch (IOException e) {
      System.out.println("An error occurred while appending to the file.");
      e.printStackTrace();
    }
  }
}
```

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 FileStats {
  public static void main(String[] args) {
    String path = "hello.txt";
    int lineCount = 0;
    int wordCount = 0;
    int charCount = 0;

    try {
      BufferedReader reader = new BufferedReader(new FileReader(path));
      String line;

      while ((line = reader.readLine()) != null) {
        lineCount++;
        String[] words = line.split("\\s+");
        wordCount += words.length;
        charCount += line.length();
      }

      reader.close();

      System.out.println("Lines: " + lineCount);
      System.out.println("Words: " + wordCount);
      System.out.println("Characters: " + charCount);
    } catch (IOException e) {
      System.out.println("An error occurred while reading the file.");
      e.printStackTrace();
    }
  }
}
```

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) {
    String sourcePath = "source.txt";
    String destinationPath = "destination.txt";

    try {
      FileReader reader = new FileReader(sourcePath);
      FileWriter writer = new FileWriter(destinationPath);

      int ch;
      while ((ch = reader.read()) != -1) {
        writer.write(ch);
      }

      reader.close();
      writer.close();

      System.out.println("File copied successfully.");
    } catch (IOException e) {
      System.out.println("An error occurred during file copy.");
      e.printStackTrace();
    }
  }
}
```

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

```
import java.io.File;

public class ListFilesInDirectory {
  public static void main(String[] args) {
    String directoryPath = "your_directory_path_here";
    File directory = new File(directoryPath);

    if (directory.exists() && directory.isDirectory()) {
      File[] filesList = directory.listFiles();

      if (filesList != null && filesList.length > 0) {
        System.out.println("Files in the directory:");
        for (File file : filesList) {
          System.out.println(file.getName());
        }
      } else {
        System.out.println("The directory is empty.");
      }
    } else {
```

```
        System.out.println("The specified path is not a valid directory.");
      }
    }
}
```

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 TxtFileFilter {
    public static void main(String[] args) {
        String directoryPath = "your_directory_path_here"; // Replace with the actual path
        File directory = new File(directoryPath);

        if (directory.exists() && directory.isDirectory()) {
            FilenameFilter txtFilter = new FilenameFilter() {
                public boolean accept(File dir, String name) {
                    return name.toLowerCase().endsWith(".txt");
                }
            };

            String[] txtFiles = directory.list(txtFilter);

            if (txtFiles != null && txtFiles.length > 0) {
                System.out.println(".txt files in the directory:");
                for (String fileName : txtFiles) {
                    System.out.println(fileName);
                }
            } else {
                System.out.println("No .txt files found in the directory.");
            }
        } else {
            System.out.println("The specified path is not a valid directory.");
        }
    }
}
```

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

```
import java.io.Serializable;

public class Student implements Serializable {
    private static final long serialVersionUID = 1L;

    private String name;
    private int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }
```

```
  public String toString() {
    return "Student{name='" + name + "', age=" + age + "}";
  }
}


import java.io.*;

public class SerializeDeserializeStudent {
  public static void main(String[] args) {
    Student student = new Student("Alice", 21);
    String filename = "student.ser";

    // Serialize the student object
    try (FileOutputStream fileOut = new FileOutputStream(filename);
         ObjectOutputStream out = new ObjectOutputStream(fileOut)) {
      out.writeObject(student);
      System.out.println("Student object serialized to " + filename);
    } catch (IOException e) {
      e.printStackTrace();
    }

    // Deserialize the student object
    try (FileInputStream fileIn = new FileInputStream(filename);
         ObjectInputStream in = new ObjectInputStream(fileIn)) {
      Student deserializedStudent = (Student) in.readObject();
      System.out.println("Deserialized Student: " + deserializedStudent);
    } catch (IOException | ClassNotFoundException e) {
      e.printStackTrace();
    }
  }
}
```

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

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

public class ReadFileWithScanner {
  public static void main(String[] args) {
    String filePath = "hello.txt"; // Replace with your file path

    try {
      Scanner scanner = new Scanner(new File(filePath));

      while (scanner.hasNext()) {
        String token = scanner.next();
        System.out.println(token);
      }
```

```
            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + filePath);
            e.printStackTrace();
        }
    }
}
```

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

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

public class WordCounter {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the filename: ");
        String filename = input.nextLine();
        System.out.print("Enter the word to search for: ");
        String targetWord = input.nextLine().toLowerCase();
        int count = 0;

        try {
            Scanner fileScanner = new Scanner(new File(filename));
            while (fileScanner.hasNext()) {
                String word = fileScanner.next().toLowerCase();
                if (word.equals(targetWord)) {
                    count++;
                }
            }
            fileScanner.close();
            System.out.println("The word '" + targetWord + "' appears " + count + " times in the file '" +
filename + "'.");
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + filename);
        }
    }
}
```

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

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

public class FileOperations {
    public static void main(String[] args) {
        Path originalPath = Paths.get("example.txt");
        Path targetPath = Paths.get("moved/example.txt");

        try {
```

```
        // Create the file
        if (!Files.exists(originalPath)) {
            Files.createFile(originalPath);
            System.out.println("File created: " + originalPath);
        }

        // Create target directory if it doesn't exist
        if (!Files.exists(targetPath.getParent())) {
            Files.createDirectories(targetPath.getParent());
        }

        // Move the file
        Files.move(originalPath, targetPath, StandardCopyOption.REPLACE_EXISTING);
        System.out.println("File moved to: " + targetPath);

        // Delete the file
        Files.delete(targetPath);
        System.out.println("File deleted: " + targetPath);

    } catch (IOException e) {
        System.out.println("An error occurred: " + e.getMessage());
    }
  }
}
```

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

```
import java.io.IOException;
import java.nio.file.*;
import java.util.List;

public class ReadFileExample {
    public static void main(String[] args) {
        Path filePath = Paths.get("example.txt");

        try {
            List<String> lines = Files.readAllLines(filePath);
            for (String line : lines) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("Error reading the file: " + e.getMessage());
        }
    }
}
```

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

StandardOpenOption.APPEND

```
import java.io.IOException;
```

```
import java.nio.file.*;
import java.util.Arrays;

public class WriteAppendExample {
  public static void main(String[] args) {
    Path filePath = Paths.get("output.txt");

    try {

      Files.write(filePath, Arrays.asList("This is the first line."), StandardOpenOption.CREATE,
StandardOpenOption.TRUNCATE_EXISTING);


      Files.write(filePath, Arrays.asList("This is an appended line."), StandardOpenOption.APPEND);

      System.out.println("Data written and appended successfully.");
    } catch (IOException e) {
      System.out.println("Error: " + e.getMessage());
    }
  }
}
```

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

```
import java.io.IOException;
import java.nio.file.*;
import java.util.stream.Stream;

public class DirectoryWalker {
  public static void main(String[] args) {
    Path startPath = Paths.get("your_directory_here");

    try (Stream<Path> paths = Files.walk(startPath)) {
      paths.filter(Files::isRegularFile)
          .forEach(System.out::println);
    } catch (IOException e) {
      System.out.println("Error walking the directory: " + e.getMessage());
    }
  }
}
```

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

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

public class FileCopyExample {
  public static void main(String[] args) {
    Path sourcePath = Paths.get("source.txt");
    Path destinationPath = Paths.get("destination.txt");
```

```
        try {
            Files.copy(sourcePath, destinationPath, StandardCopyOption.REPLACE_EXISTING);
            System.out.println("File copied successfully.");
        } catch (IOException e) {
            System.out.println("Error copying file: " + e.getMessage());
        }
    }
}
```

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

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

public class FileSizeExample {
    public static void main(String[] args) {
        Path filePath = Paths.get("example.txt");

        try {
            long size = Files.size(filePath);
            System.out.println("File size: " + size + " bytes");
        } catch (IOException e) {
            System.out.println("Error getting file size: " + e.getMessage());
        }
    }
}
```

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

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.Serializable;

class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    String name;
    int id;
    double salary;

    public Employee(String name, int id, double salary) {
        this.name = name;
        this.id = id;
        this.salary = salary;
    }
}

public class SerializeEmployee {
```

```
        public static void main(String[] args) {
            Employee emp = new Employee("John Doe", 101, 50000.0);

            try (FileOutputStream fileOut = new FileOutputStream("employee.ser");
                 ObjectOutputStream out = new ObjectOutputStream(fileOut)) {
                out.writeObject(emp);
                System.out.println("Serialized data is saved in employee.ser");
            } catch (IOException e) {
                System.out.println("Serialization error: " + e.getMessage());
            }
        }
    }
```

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

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.Serializable;

class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    String name;
    int id;
    double salary;
}

public class DeserializeEmployee {
    public static void main(String[] args) {
        try (FileInputStream fileIn = new FileInputStream("employee.ser");
             ObjectInputStream in = new ObjectInputStream(fileIn)) {

            Employee emp = (Employee) in.readObject();
            System.out.println("Deserialized Employee Data:");
            System.out.println("Name: " + emp.name);
            System.out.println("ID: " + emp.id);
            System.out.println("Salary: " + emp.salary);
```

```
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Deserialization error: " + e.getMessage());
        }
    }
}
```