Task 1: Create a File and Write Text

Explanation:

- Use `FileWriter` and `BufferedWriter` to write to a file.
- `true` in `FileWriter` constructor enables appending.

□ Solution:

```
import java.io.*;

public class Task1 {
    public static void main(String[] args) {
        try {
            BufferedWriter writer = new BufferedWriter(new
FileWriter("info.txt", true));
            writer.write("Java I/O is powerful!\n");
            writer.write("Learn Java step-by-step\n");
            writer.close();
            System.out.println("File written successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

□ Task 2: Read from the File Created in Task 1

Explanation:

- `BufferedReader.readLine()` reads lines until `null`.

□ Solution:

```
import java.io.*;

public class Task2 {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new
FileReader("info.txt"));
            String line;
            while ((line = reader.readLine()) != null)
                System.out.println(line);
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

□ Task 3: Count the Number of Words in a File

Explanation:

- Use `split("\\s+")` to split by space(s) and count words.

□ Solution:

```
import java.io.*;

public class Task3 {
    public static void main(String[] args) {
        int wordCount = 0;
        try {
            BufferedReader reader = new BufferedReader(new
FileReader("info.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.trim().split("\\s+");
                wordCount += words.length;
            }
            reader.close();
            System.out.println("Total words: " + wordCount);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

□ Task 4: Copy Content from One File to Another

Explanation:

- Read from one file, write to another using `BufferedReader` and `BufferedWriter`.

□ Solution:

```
import java.io.*;

public class Task4 {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new
FileReader("info.txt"));
            BufferedWriter writer = new BufferedWriter(new
FileWriter("copy.txt"));

            String line;
            while ((line = reader.readLine()) != null)
                writer.write(line + "\n");
            reader.close();
            writer.close();
            System.out.println("File copied.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

□ Task 5: Write and Read Student Details (Serialization)

Explanation:

- `ObjectOutputStream` serializes, `ObjectInputStream` deserializes.

□ Solution:

```
import java.io.*;

class Student implements Serializable {
    String name;
    int rollNo;
    int marks;

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

public class Task5 {
    public static void main(String[] args) throws Exception {
        Student s = new Student("Rahul", 101, 85);
        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("student.ser"));
        oos.writeObject(s);
        oos.close();

        ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("student.ser"));
        Student obj = (Student) ois.readObject();
        ois.close();

        System.out.println("Deserialized Student:\nName: " + obj.name + ",
Roll No: " + obj.rollNo + ", Marks: " + obj.marks);
    }
}
```

□ Task 6: Take Input from Keyboard and Save to File

Explanation:

- Use `Scanner` for input, `BufferedWriter` to write.

□ Solution:

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

public class Task6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            BufferedWriter writer = new BufferedWriter(new
FileWriter("userdata.txt"));
            System.out.print("Enter Name: ");
            writer.write("Name: " + sc.nextLine() + "\n");
            System.out.print("Enter Email: ");
            writer.write("Email: " + sc.nextLine() + "\n");
            System.out.print("Enter Address: ");
            writer.write("Address: " + sc.nextLine() + "\n");
            writer.close();
            System.out.println("Data saved.");
        } catch (IOException e) {
            e.printStackTrace();
        }
        sc.close();
    }
}
```

□ Task 7: Create a Log File with Timestamps

Explanation:

- Use `Date` and `SimpleDateFormat` to log timestamp.

□ Solution:

```
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Task7 {
    public static void main(String[] args) {
        try {
            BufferedWriter writer = new BufferedWriter(new
FileWriter("app.log", true));
```

```
            String timestamp = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").format(new Date());
            writer.write("[" + timestamp + "] Application started\n");
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

□ Task 8: Contact Manager Using File Storage

Solution (Simplified):

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

public class Task8 {
    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        BufferedWriter writer = new BufferedWriter(new
FileWriter("contacts.txt", true));

        System.out.print("Enter name: ");
        String name = sc.nextLine();
        System.out.print("Enter phone: ");
        String phone = sc.nextLine();
        writer.write(name + " - " + phone + "\n");
        writer.close();

        System.out.println("All Contacts:");
        BufferedReader reader = new BufferedReader(new
FileReader("contacts.txt"));
        String line;
        while ((line = reader.readLine()) != null)
            System.out.println(line);
        reader.close();
        sc.close();
    }
}
```

□ Task 9: Count Lines, Words, and Characters

□ Solution:

```
import java.io.*;

public class Task9 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader("sample.txt"));
        int lines = 0, words = 0, chars = 0;
        String line;

        while ((line = br.readLine()) != null) {
            lines++;
            String[] wordList = line.split("\\s+");
            words += wordList.length;
            chars += line.replace(" ", "").length();
        }
        br.close();

        System.out.println("Lines: " + lines);
        System.out.println("Words: " + words);
        System.out.println("Characters (no spaces): " + chars);
    }
}
```

□ Task 10: Store and Retrieve Multiple Students

□ Solution:

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

class Student implements Serializable {
    String name;
    int roll;
    Student(String n, int r) {
        name = n;
        roll = r;
    }
}

public class Task10 {
    public static void main(String[] args) throws Exception {
        ArrayList<Student> list = new ArrayList<>();
        list.add(new Student("A", 1));
        list.add(new Student("B", 2));

        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("students.ser"));
        oos.writeObject(list);
        oos.close();
        ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("students.ser"));
        ArrayList<Student> students = (ArrayList<Student>) ois.readObject();
        ois.close();

        for (Student s : students)
            System.out.println(s.name + " - " + s.roll);
    }
}
```

□ Task 11: File Not Found Exception Handling

□ Solution:

```
import java.io.*;

public class Task11 {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new
FileReader("nofile.txt"));
            reader.readLine();
        } catch (FileNotFoundException e) {
            System.out.println("File not found. Please check the file
name.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

□ Task 12: Keyboard Input and Reverse Text

□ Solution:

```
import java.util.Scanner;

public class Task12 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter text: ");
        String input = sc.nextLine();
        String reversed = new StringBuilder(input).reverse().toString();
        System.out.println("Reversed: " + reversed);
        sc.close();
    }
}
```

Task 13: Read and Replace Words in File

Solution:

```
import java.io.*;

public class Task13 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader("data.txt"));
        BufferedWriter bw = new BufferedWriter(new FileWriter("output.txt"));

        String line;
        while ((line = br.readLine()) != null)
            bw.write(line.replace("Java", "Python") + "\n");

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

Task 14: Merge Two Files

Solution:

```
import java.io.*;

public class Task14 {
    public static void main(String[] args) throws IOException {
        BufferedWriter bw = new BufferedWriter(new FileWriter("merged.txt"));

        BufferedReader br1 = new BufferedReader(new FileReader("file1.txt"));
        BufferedReader br2 = new BufferedReader(new FileReader("file2.txt"));

        String line;
        while ((line = br1.readLine()) != null) bw.write(line + "\n");
        while ((line = br2.readLine()) != null) bw.write(line + "\n");

        br1.close(); br2.close(); bw.close();
    }
}
```

Task 15: Directory Scanner

Solution:

```
import java.io.File;

public class Task15 {
    public static void main(String[] args) {
        File dir = new File(".");

        File[] files = dir.listFiles();
        for (File file : files) {
            System.out.println((file.isDirectory() ? "DIR " : "FILE") +
                    " - " + file.getName() + " - " + file.length() + "
bytes");
        }
    }
}
```