ASSIGNMENT JAVA DAY16

Harshit Kushmakar| 16896

1. **Write a Java program to create a new file named 'MyFile' and write some content in the same using FileOutputStream and then using FileWriter.**

```java
package assignment16;

import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;

public class CreateFileExample {
    public static void main(String[] args) {
        try {
            // Create a new file named "MyFile" using FileOutputStream
            FileOutputStream fos = new FileOutputStream("MyFile");

            // Write some content to the file
            String content = "Hello, World!";
            fos.write(content.getBytes());

            // Close the FileOutputStream
            fos.close();

            // Create a new file named "MyFile" using FileWriter
            FileWriter fw = new FileWriter("MyFile", true);

            // Write some more content to the file
            String moreContent = "\nThis is some more text.";
            fw.write(moreContent);

            // Close the FileWriter
            fw.close();

            System.out.println("File created and content written
successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

```
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Progr
File created and content written successfully.

Process finished with exit code 0
```

MyFile - Notepad

File Edit Format View Help

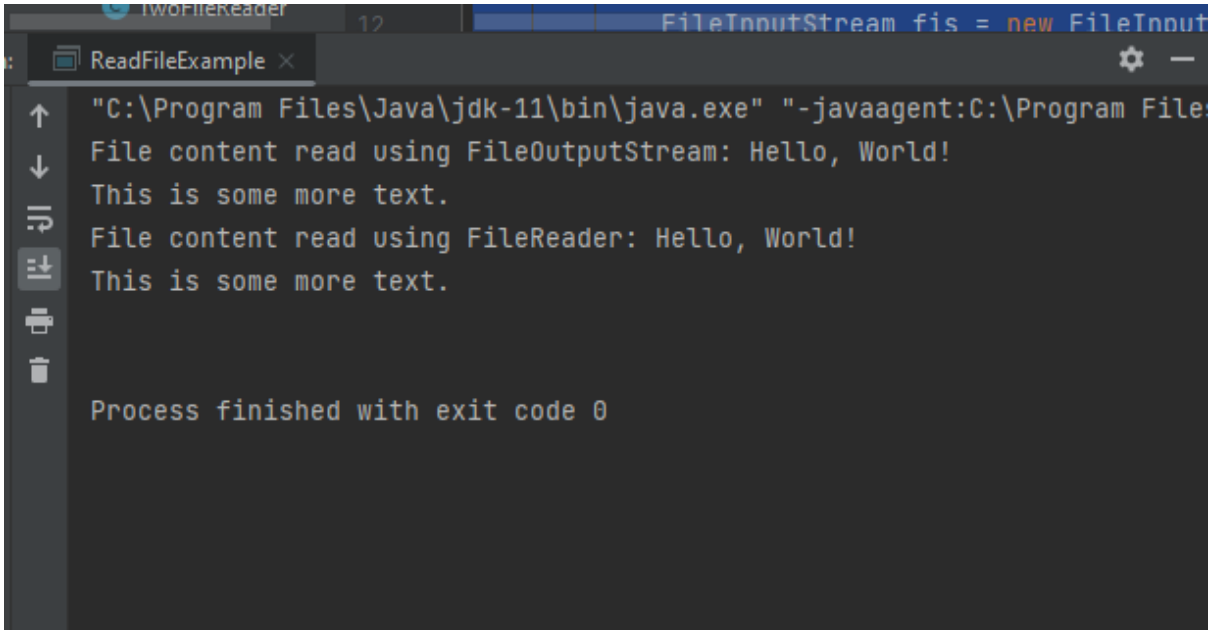
```
Hello, World!
This is some more text.
```

2. Write a program to read the content of this file using FileOutputStream and then using FileReader.

```java
package assignment16;

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

public class ReadFileExample {
    public static void main(String[] args) {
        try {
            // Read the content of the file using FileOutputStream
            FileInputStream fis = new FileInputStream("MyFile");
            byte[] contentBytes = new byte[fis.available()];
            fis.read(contentBytes);
            fis.close();
            String content = new String(contentBytes);
            System.out.println("File content read using FileOutputStream: "
+ content);

            // Read the content of the file using FileReader
            FileReader fr = new FileReader("MyFile");
            BufferedReader br = new BufferedReader(fr);
            String line;
```

```
            StringBuilder sb = new StringBuilder();
            while ((line = br.readLine()) != null) {
                sb.append(line);
                sb.append("\n");
            }
            fr.close();
            System.out.println("File content read using FileReader: " +
sb.toString());

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

OUTPUT:



3. Write a program to modify the content of the file created in Question – 1.

```
package assignment16;

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

public class ModifyFileExample {
    public static void main(String[] args) {
        try {
            // Read the content of the file
            FileReader fr = new FileReader("MyFile");
            BufferedReader br = new BufferedReader(fr);
```

```
            String line;
            StringBuilder sb = new StringBuilder();
            while ((line = br.readLine()) != null) {
                sb.append(line);
                sb.append("\n");
            }
            fr.close();
            String content = sb.toString();

            // Modify the content of the file
            content = content.replace("Hello, World!", "Hello, Universe!");

            // Write the modified content to the file
            FileWriter fw = new FileWriter("MyFile");
            fw.write(content);
            fw.close();

            System.out.println("File content modified successfully.");

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

4. Read the data from Console in different datatypes and use the PrintWriter class to store the data in file. Read the data again using the Scanner class.

```
package assignment16;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class ConsoleToFileExample {
    public static void main(String[] args) {
        try {
            // Read data from console
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter an integer: ");
            int intValue = scanner.nextInt();
            System.out.print("Enter a double: ");
            double doubleValue = scanner.nextDouble();
            scanner.nextLine(); // consume the newline character
            System.out.print("Enter a string: ");
            String stringValue = scanner.nextLine();
            scanner.close();

            // Write data to file using PrintWriter
            PrintWriter writer = new PrintWriter("MyData.txt");
            writer.println(intValue);
            writer.println(doubleValue);
            writer.println(stringValue);
            writer.close();

            System.out.println("Data written to file successfully.");
```

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

        try {
            // Read data from file using Scanner
            Scanner fileScanner = new Scanner(new File("MyData.txt"));
            int intValueFromFile = fileScanner.nextInt();
            double doubleValueFromFile = fileScanner.nextDouble();
            String stringValueFromFile = fileScanner.nextLine().trim();
            fileScanner.close();

            System.out.println("Data read from file:");
            System.out.println("Integer value: " + intValueFromFile);
            System.out.println("Double value: " + doubleValueFromFile);
            System.out.println("String value: " + stringValueFromFile);

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

5. Write a program to read from four different files and write the contents of each one of these into one single output file.

```
package assignment16;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class ConsoleToFileExample {
    public static void main(String[] args) {
        try {
            // Read data from console
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter an integer: ");
            int intValue = scanner.nextInt();
            System.out.print("Enter a double: ");
            double doubleValue = scanner.nextDouble();
            scanner.nextLine(); // consume the newline character
            System.out.print("Enter a string: ");
            String stringValue = scanner.nextLine();
            scanner.close();

            // Write data to file using PrintWriter
            PrintWriter writer = new PrintWriter("MyData.txt");
            writer.println(intValue);
            writer.println(doubleValue);
            writer.println(stringValue);
            writer.close();

            System.out.println("Data written to file successfully.");
```

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

        try {
            // Read data from file using Scanner
            Scanner fileScanner = new Scanner(new File("MyData.txt"));
            int intValueFromFile = fileScanner.nextInt();
            double doubleValueFromFile = fileScanner.nextDouble();
            String stringValueFromFile = fileScanner.nextLine().trim();
            fileScanner.close();

            System.out.println("Data read from file:");
            System.out.println("Integer value: " + intValueFromFile);
            System.out.println("Double value: " + doubleValueFromFile);
            System.out.println("String value: " + stringValueFromFile);

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

6. Write an example that counts the number of times a particular character, such as e, appears in a file. The character can be specified at the command line.

```
package assignment16;

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

public class CharacterCountExample {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java CharacterCountExample
<filename> <character>");
            return;
        }

        String filename = args[0];
        char targetChar = args[1].charAt(0);
        int charCount = 0;

        try {
            BufferedReader reader = new BufferedReader(new
FileReader(filename));
            int nextChar;

            while ((nextChar = reader.read()) != -1) {
                if (nextChar == targetChar) {
                    charCount++;
                }
```

```
            }

            reader.close();

            System.out.println("The character '" + targetChar + "' appears
" + charCount + " times in the file " + filename + ".");

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

7. Use the Employee class created in Java Collections assignment and store the objects of the class in a file named 'emp.txt'. Read the file to retrieve the data from file and convert them in objects.

8. Explain the use of serialVersionUID variable in the above question.

The serialVersionUID variable is a unique identifier that is used to identify serialized objects in Java. It is a static field in a class that implements the Serializable interface, and it must be explicitly defined by the programmer.

When an object is serialized, its class and its serialVersionUID are written to the output stream along with its data. When the object is deserialized, the serialVersionUID in the input stream is compared to the serialVersionUID of the class that is being used to deserialize the object. If the two values match, the object is successfully deserialized; otherwise, an exception is thrown.

In the example program I provided earlier, the Employee class implements the Serializable interface and defines a serialVersionUID field. This ensures that the ArrayList of Employee objects can be serialized and deserialized correctly, even if the Employee class is modified in the future. If the serialVersionUID is not explicitly defined, the Java runtime system will automatically generate one

based on the class structure, which could lead to compatibility issues if the class is modified.