

Day 20 Assignment

Task 1: Java IO Basics

Write a program that reads a text file and counts the frequency of each word using `FileReader` and `FileWriter`.

```
WordFrequencyCounter.java x input.txt output.txt
1 package assignment;
2
3 import java.io.*;
4 import java.util.HashMap;
5 import java.util.Map;
6 import java.util.Scanner;
7
8 public class WordFrequencyCounter {
9     public static void main(String[] args) {
10         if (args.length != 2) {
11             System.out.println("Usage: java WordFrequencyCounter <inputFile> <outputFile>");
12             return;
13         }
14
15         String inputFileName = args[0];
16         String outputFileName = args[1];
17
18         Map<String, Integer> wordCounts = new HashMap<>();
19
20         try (FileReader fileReader = new FileReader(inputFileName);
21             Scanner scanner = new Scanner(fileReader)) {
22
23             while (scanner.hasNext()) {
24                 String word = scanner.next().toLowerCase().replaceAll("[^a-zA-Z]", "");
25                 if (!word.isEmpty()) {
26                     wordCounts.put(word, wordCounts.getOrDefault(word, 0) + 1);
27                 }
28             }
29         } catch (IOException e) {
30             System.out.println("An error occurred while reading the file: " + e.getMessage());
31         }
32
33         try (FileWriter fileWriter = new FileWriter(outputFileName)) {
34             for (Map.Entry<String, Integer> entry : wordCounts.entrySet()) {
35                 fileWriter.write(entry.getKey() + ": " + entry.getValue() + System.lineSeparator());
36             }
37         } catch (IOException e) {
38             System.out.println("An error occurred while writing to the file: " + e.getMessage());
39         }
40
41         System.out.println("Word frequency count has been written to " + outputFileName);
42     }
43 }
```

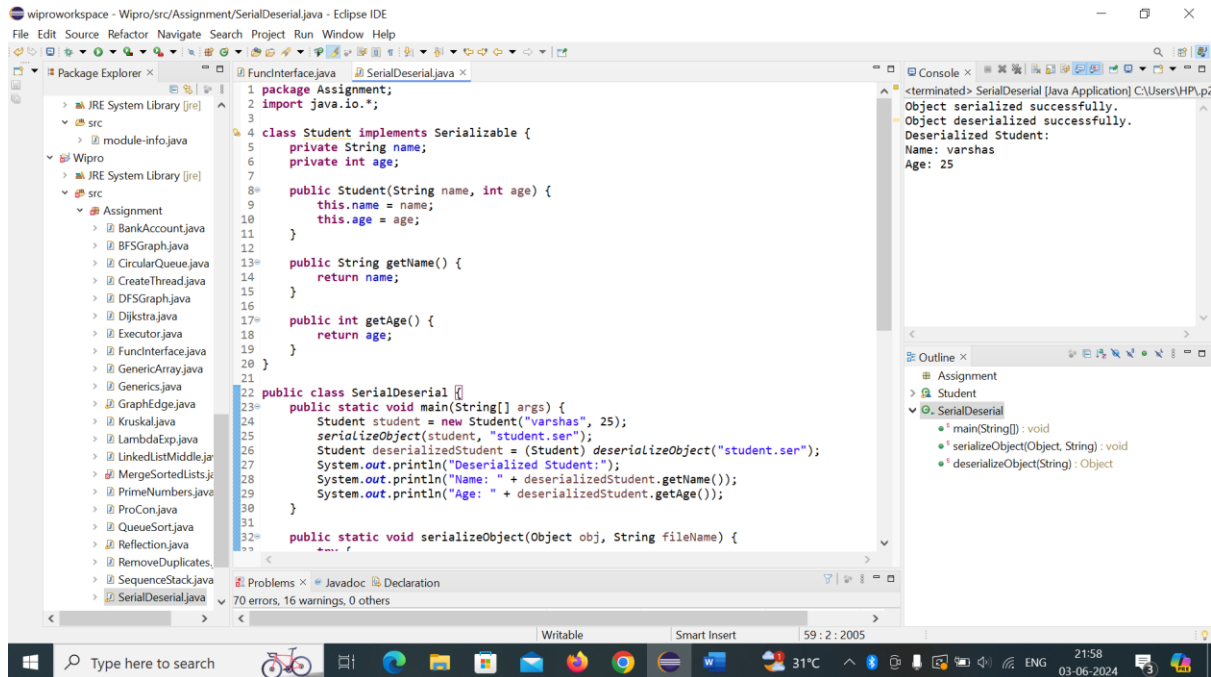
```
<terminated> WordFrequencyCounter [Java Application] C:\Program Fil
Word frequency count has been written to output.txt
```

```
WordFrequencyCounter.java input.txt x output.txt
1 Hi My name is varsha.
```

```
WordFrequencyCounter.java input.txt output.txt x
1 hi: 1
2 name: 1
3 is: 1
4 my: 1
5 varsha: 1
6
```

Task 2: Serialization and Deserialization

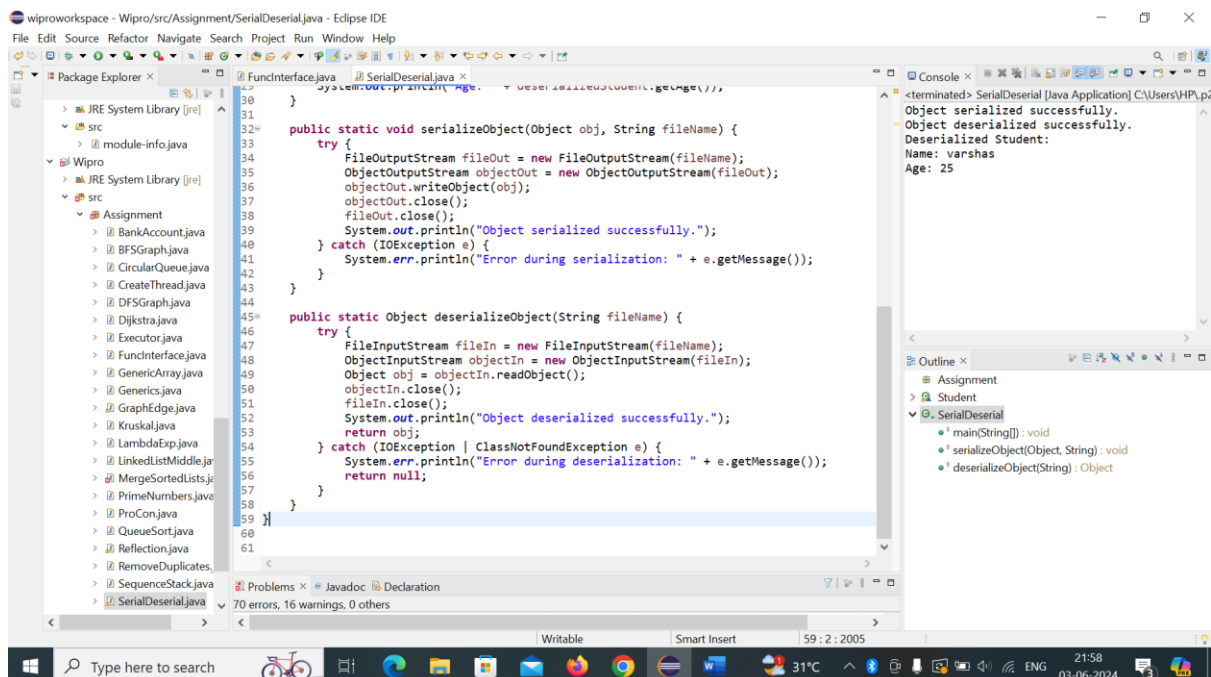
Serialize a custom object to a file and then deserialize it back to recover the object state.



```
1 package Assignment;
2 import java.io.*;
3
4 class Student implements Serializable {
5     private String name;
6     private int age;
7
8     public Student(String name, int age) {
9         this.name = name;
10        this.age = age;
11    }
12
13    public String getName() {
14        return name;
15    }
16
17    public int getAge() {
18        return age;
19    }
20 }
21
22 public class SerialDeserial {
23     public static void main(String[] args) {
24         Student student = new Student("varshas", 25);
25         serializeObject(student, "student.ser");
26         Student deserializedStudent = (Student) deserializeObject("student.ser");
27         System.out.println("Deserialized Student:");
28         System.out.println("Name: " + deserializedStudent.getName());
29         System.out.println("Age: " + deserializedStudent.getAge());
30     }
31
32     public static void serializeObject(Object obj, String fileName) {
33         // ...
34     }
35
36     public static Object deserializeObject(String fileName) {
37         // ...
38     }
39 }
```

Console output:

```
<terminated> SerialDeserial [Java Application] C:\Users\VHP.p...
Object serialized successfully.
Object deserialized successfully.
Deserialized Student:
Name: varshas
Age: 25
```



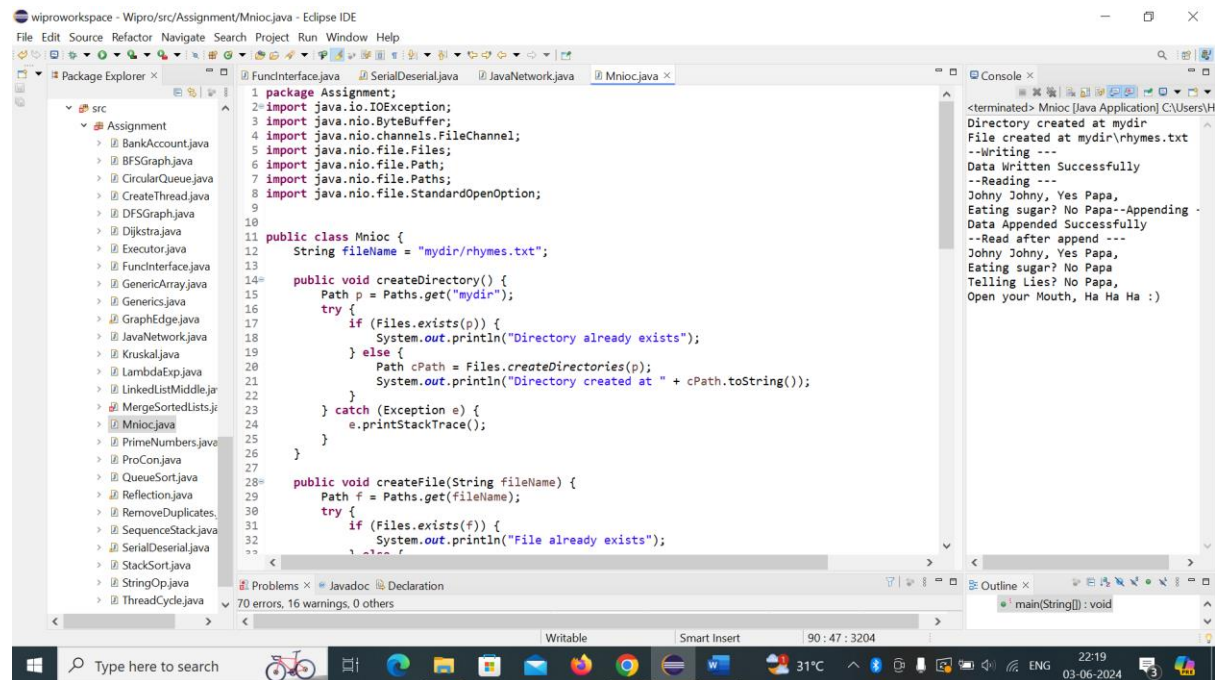
```
30 }
31
32 public static void serializeObject(Object obj, String fileName) {
33     try {
34         FileOutputStream fileOut = new FileOutputStream(fileName);
35         ObjectOutputStream objectOut = new ObjectOutputStream(fileOut);
36         objectOut.writeObject(obj);
37         objectOut.close();
38         fileOut.close();
39         System.out.println("Object serialized successfully.");
40     } catch (IOException e) {
41         System.err.println("Error during serialization: " + e.getMessage());
42     }
43 }
44
45 public static Object deserializeObject(String fileName) {
46     try {
47         FileInputStream fileIn = new FileInputStream(fileName);
48         ObjectInputStream objectIn = new ObjectInputStream(fileIn);
49         Object obj = objectIn.readObject();
50         objectIn.close();
51         fileIn.close();
52         System.out.println("Object deserialized successfully.");
53         return obj;
54     } catch (IOException | ClassNotFoundException e) {
55         System.err.println("Error during deserialization: " + e.getMessage());
56         return null;
57     }
58 }
59 }
60
61 }
```

Console output:

```
<terminated> SerialDeserial [Java Application] C:\Users\VHP.p...
Object serialized successfully.
Object deserialized successfully.
Deserialized Student:
Name: varshas
Age: 25
```

Task 3: New IO (NIO)

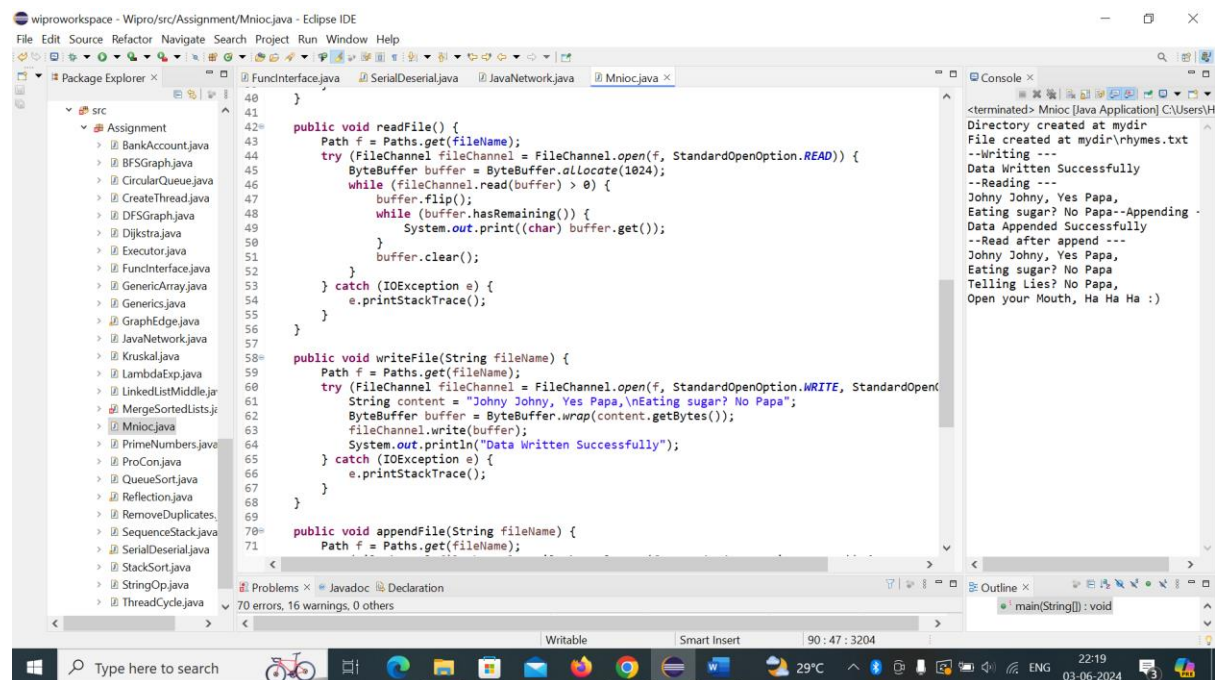
Use NIO Channels and Buffers to read content from a file and write to another file.



The screenshot shows the Eclipse IDE with the file `Mnioc.java` open. The code includes the following package, imports, and class definition:

```
1 package Assignment;
2 import java.io.IOException;
3 import java.nio.ByteBuffer;
4 import java.nio.channels.FileChannel;
5 import java.nio.file.Files;
6 import java.nio.file.Paths;
7 import java.nio.file.StandardOpenOption;
8
9
10 public class Mnioc {
11     String fileName = "mydir/rhymes.txt";
12
13     public void createDirectory() {
14         Path p = Paths.get("mydir");
15         try {
16             if (Files.exists(p)) {
17                 System.out.println("Directory already exists");
18             } else {
19                 Path cPath = Files.createDirectories(p);
20                 System.out.println("Directory created at " + cPath.toString());
21             } catch (Exception e) {
22                 e.printStackTrace();
23             }
24         }
25     }
26
27     public void createFile(String fileName) {
28         Path f = Paths.get(fileName);
29         try {
30             if (Files.exists(f)) {
31                 System.out.println("File already exists");
32             }
33         }
34     }
35 }
```

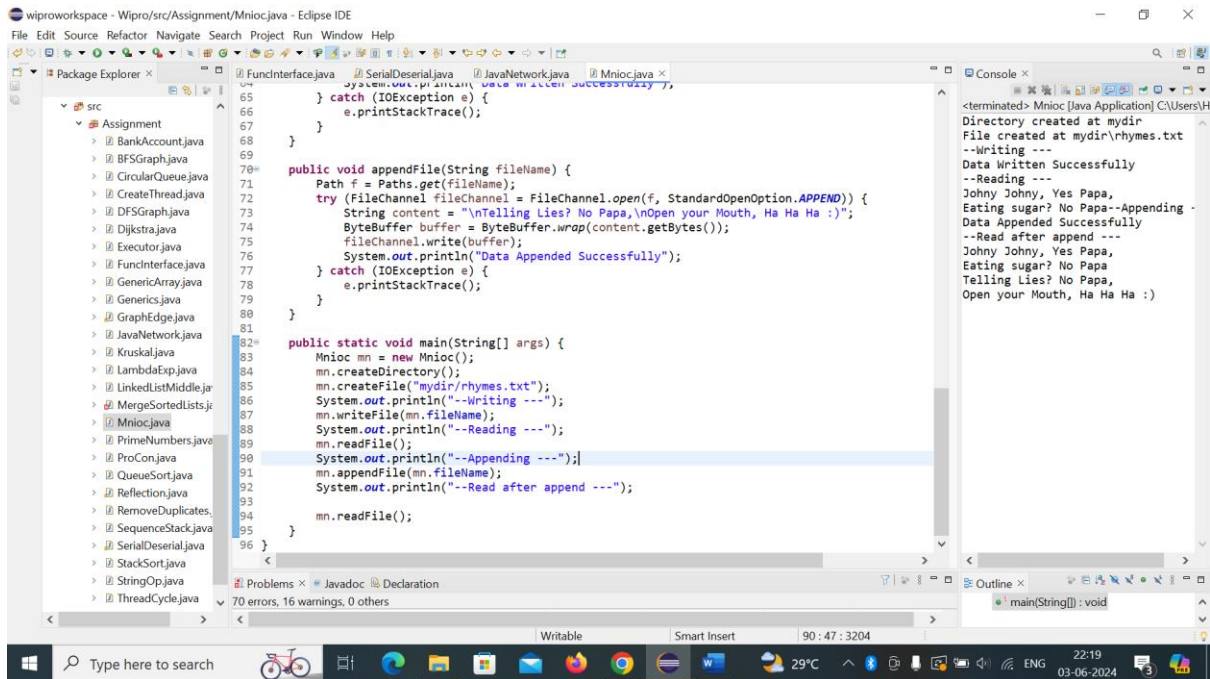
The Package Explorer on the left shows the project structure with the `src` folder containing the `Assignment` package and various Java files. The Console on the right shows the output of the program, indicating that the directory `mydir` was created and the file `rhymes.txt` was written successfully.



The screenshot shows the Eclipse IDE with the file `Mnioc.java` open. The code includes the following package, imports, and class definition:

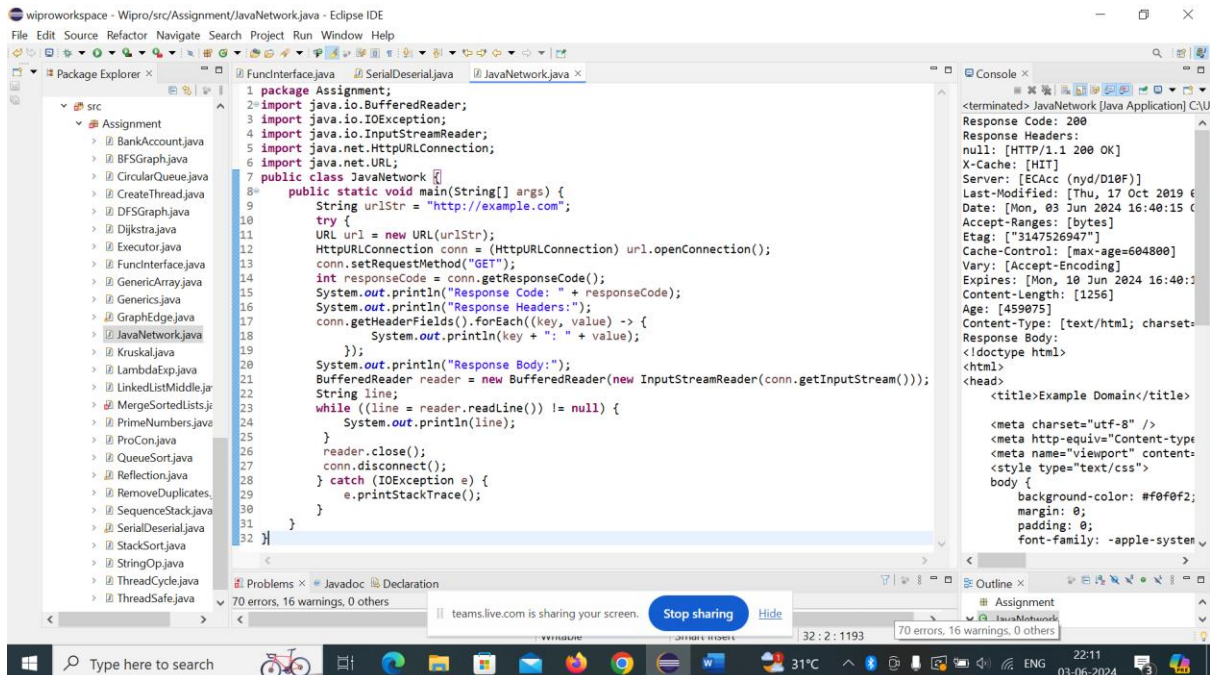
```
40 }
41
42 public void readFile() {
43     Path f = Paths.get(fileName);
44     try (FileChannel fileChannel = FileChannel.open(f, StandardOpenOption.READ)) {
45         ByteBuffer buffer = ByteBuffer.allocate(1024);
46         while (fileChannel.read(buffer) > 0) {
47             buffer.flip();
48             while (buffer.hasRemaining()) {
49                 System.out.print((char) buffer.get());
50             }
51             buffer.clear();
52         } catch (IOException e) {
53             e.printStackTrace();
54         }
55     }
56
57     public void writeFile(String fileName) {
58         Path f = Paths.get(fileName);
59         try (FileChannel fileChannel = FileChannel.open(f, StandardOpenOption.WRITE, StandardOpenOption.CREATE)) {
60             String content = "Johnny Johnny, Yes Papa,\nEating sugar? No Papa";
61             ByteBuffer buffer = ByteBuffer.wrap(content.getBytes());
62             fileChannel.write(buffer);
63             System.out.println("Data Written Successfully");
64         } catch (IOException e) {
65             e.printStackTrace();
66         }
67     }
68
69     public void appendFile(String fileName) {
70         Path f = Paths.get(fileName);
71     }
72 }
```

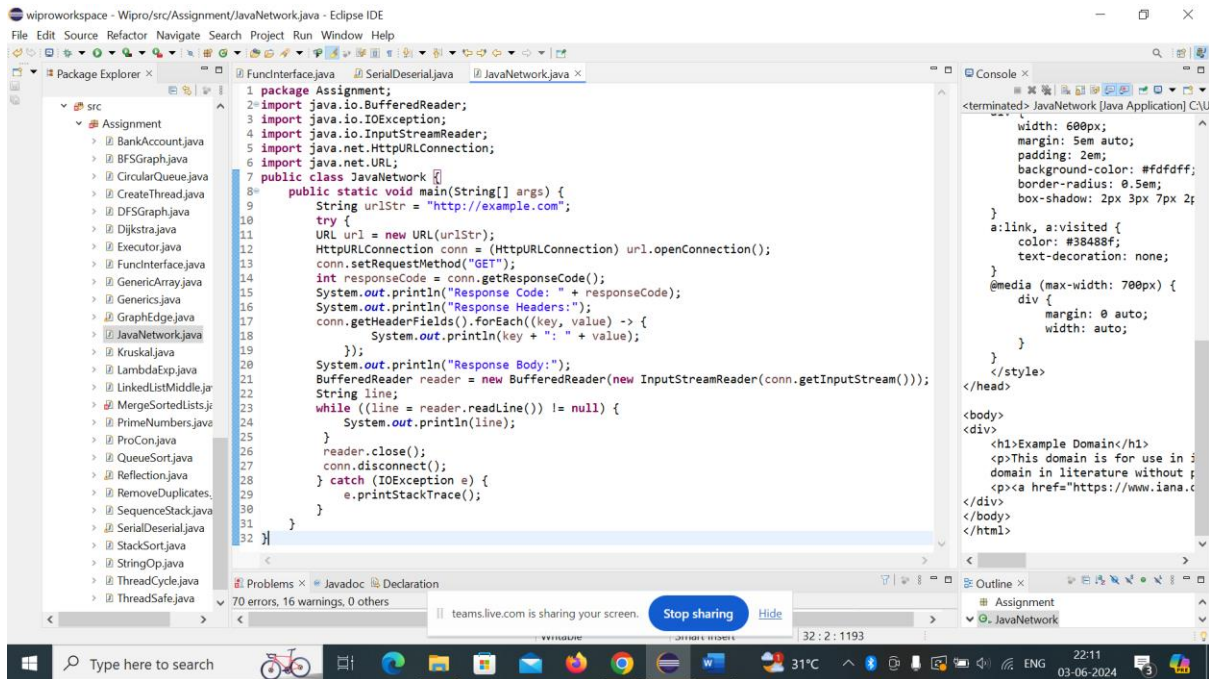
The Package Explorer on the left shows the project structure with the `src` folder containing the `Assignment` package and various Java files. The Console on the right shows the output of the program, indicating that the directory `mydir` was created and the file `rhymes.txt` was written successfully.



Task 4: Java Networking

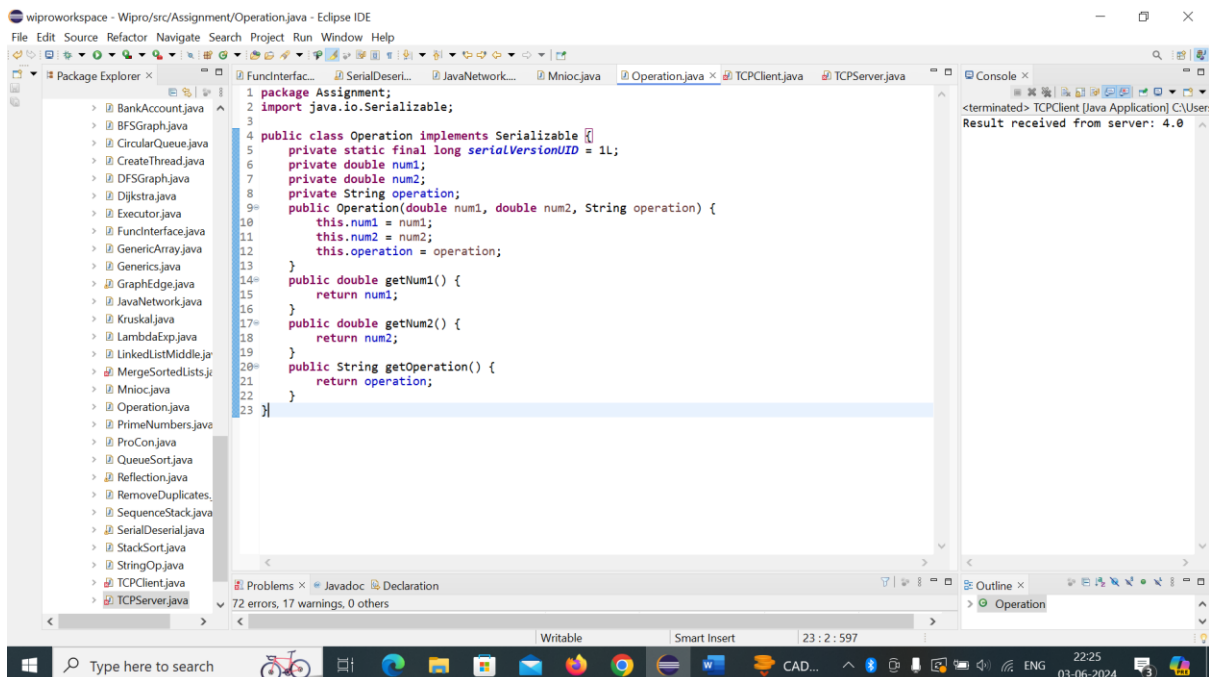
Write a simple HTTP client that connects to a URL, sends a request, and displays the response headers and body.





Task 5: Java Networking and Serialization

Develop a basic TCP client and server application where the client sends a serialized object with 2 numbers and operation to be performed on them to the server, and the server computes the result and sends it back to the client. for eg, we could send 2, 2, "+" which would mean 2 + 2



wiproworkspace - Wipro/src/Assignment/TCPClient.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- BankAccount.java
- BFSGraph.java
- CircularQueue.java
- CreateThread.java
- DFSGraph.java
- Dijkstra.java
- Executor.java
- FuncInterface.java
- GenericArray.java
- Generics.java
- GraphEdge.java
- JavaNetwork.java
- Kruskal.java
- LambdaExp.java
- LinkedListMiddle.java
- MergeSortedLists.java
- MnIOC.java
- Operation.java
- PrimeNumbers.java
- ProCon.java
- QueueSort.java
- Reflection.java
- RemoveDuplicates.java
- SequenceStack.java
- SerialDeserial.java
- StackSort.java
- StringOp.java
- TCPClient.java
- TCPServer.java

```
1 package Assignment;
2 import java.io.*;
3 import java.net.*;
4 import assignments.day20.Operation;
5 public class TCPClient {
6     public static void main(String[] args) {
7         try {
8             Socket socket = new Socket("localhost", 1234);
9             ObjectOutputStream objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
10            ObjectInputStream objectInputStream = new ObjectInputStream(socket.getInputStream());
11
12            Operation operation = new Operation(2, 2, "+");
13            objectOutputStream.writeObject(operation);
14            objectOutputStream.flush();
15
16            double result = objectInputStream.readDouble();
17            System.out.println("Result received from server: " + result);
18
19            objectOutputStream.close();
20            objectInputStream.close();
21            socket.close();
22        } catch (IOException e) {
23            e.printStackTrace();
24        }
25    }
26 }
```

Problems x Javadoc Declaration

72 errors, 17 warnings, 0 others

Writable Smart Insert 1:20:19

Type here to search 29°C 03-06-2024

wiproworkspace - Wipro/src/Assignment/TCPServer.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- BankAccount.java
- BFSGraph.java
- CircularQueue.java
- CreateThread.java
- DFSGraph.java
- Dijkstra.java
- Executor.java
- FuncInterface.java
- GenericArray.java
- Generics.java
- GraphEdge.java
- JavaNetwork.java
- Kruskal.java
- LambdaExp.java
- LinkedListMiddle.java
- MergeSortedLists.java
- MnIOC.java
- Operation.java
- PrimeNumbers.java
- ProCon.java
- QueueSort.java
- Reflection.java
- RemoveDuplicates.java
- SequenceStack.java
- SerialDeserial.java
- StackSort.java
- StringOp.java
- TCPClient.java
- TCPServer.java

```
1 package Assignment;
2 import java.io.*;
3 import java.net.*;
4 import assignments.day20.Operation;
5 public class TCPServer {
6     public static void main(String[] args) {
7         try {
8             ServerSocket serverSocket = new ServerSocket(1234);
9             System.out.println("Server started. Waiting for clients...");
10
11             while (true) {
12                 Socket socket = serverSocket.accept();
13                 System.out.println("Client connected: " + socket.getInetAddress());
14
15                 ObjectInputStream objectInputStream = new ObjectInputStream(socket.getInputStream());
16                 ObjectOutputStream objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
17
18                 Operation operation = (Operation) objectInputStream.readObject();
19                 double result = performOperation(operation);
20
21                 objectOutputStream.writeDouble(result);
22                 objectOutputStream.flush();
23
24                 objectInputStream.close();
25                 objectOutputStream.close();
26                 socket.close();
27             }
28         } catch (IOException | ClassNotFoundException e) {
29             e.printStackTrace();
30         }
31     }
32     private static double performOperation(Operation operation) {
33         double num1 = operation.getNum1();
34         double num2 = operation.getNum2();
35         String op = operation.getOp();
36         double result = 0;
37         switch (op) {
38             case "+":
39                 result = num1 + num2;
40             case "-":
41                 result = num1 - num2;
42             case "*":
43                 result = num1 * num2;
44             case "/":
45                 result = num1 / num2;
46             default:
47                 result = 0;
48         }
49         return result;
50     }
51 }
```

Problems x Javadoc Declaration

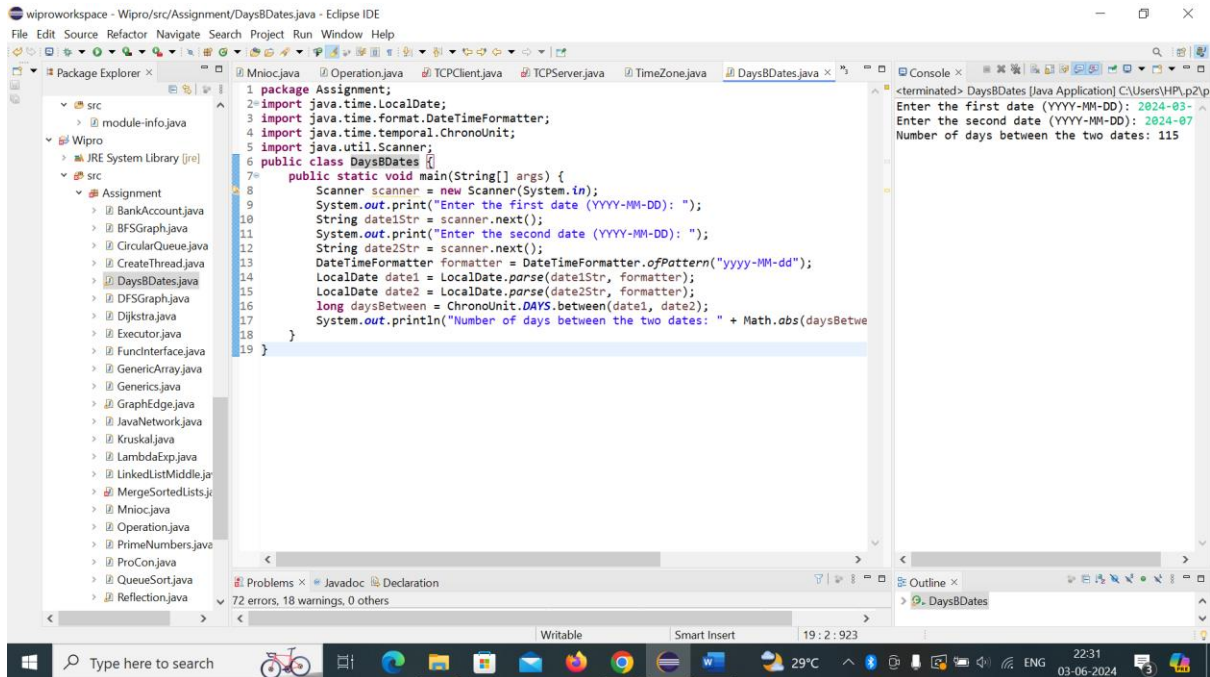
72 errors, 17 warnings, 0 others

Writable Smart Insert 53:2:1851

Type here to search 29°C 03-06-2024

Task 6: Java 8 Date and Time API

Write a program that calculates the number of days between two dates input by the user.



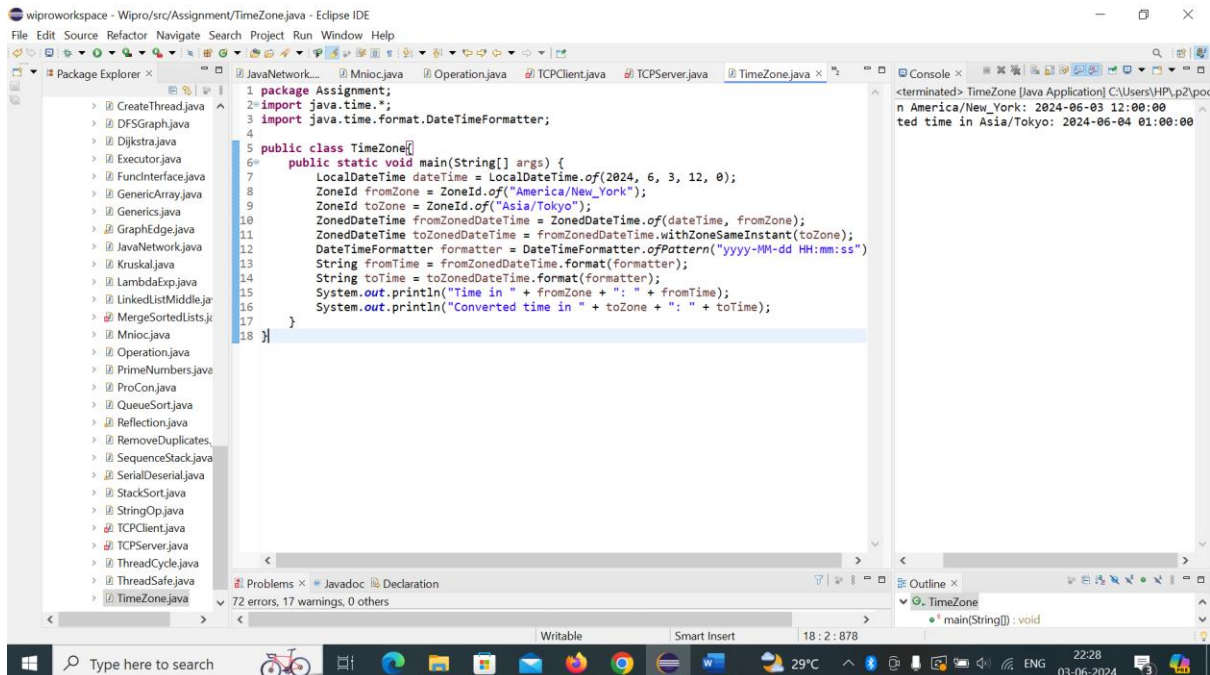
The screenshot shows the Eclipse IDE with the `DaysBDates.java` file open. The code is as follows:

```
1 package Assignment;
2 import java.time.LocalDate;
3 import java.time.format.DateTimeFormatter;
4 import java.time.temporal.ChronoUnit;
5 import java.util.Scanner;
6 public class DaysBDates {
7     public static void main(String[] args) {
8         Scanner scanner = new Scanner(System.in);
9         System.out.print("Enter the first date (YYYY-MM-DD): ");
10        String date1Str = scanner.next();
11        System.out.print("Enter the second date (YYYY-MM-DD): ");
12        String date2Str = scanner.next();
13        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
14        LocalDate date1 = LocalDate.parse(date1Str, formatter);
15        LocalDate date2 = LocalDate.parse(date2Str, formatter);
16        long daysBetween = ChronoUnit.DAYS.between(date1, date2);
17        System.out.println("Number of days between the two dates: " + Math.abs(daysBetween));
18    }
19 }
```

The console output shows the program execution:

```
<terminated> DaysBDates [Java Application] C:\Users\HP\AppData\Local\Temp\wipro-workspace\Wipro\src\Assignment\DaysBDates.java
Enter the first date (YYYY-MM-DD): 2024-03-
Enter the second date (YYYY-MM-DD): 2024-07
Number of days between the two dates: 115
```

Task 7: Timezone Create a timezone converter that takes a time in one timezone and converts it to another timezone.



The screenshot shows the Eclipse IDE with the `TimeZone.java` file open. The code is as follows:

```
1 package Assignment;
2 import java.time.*;
3 import java.time.format.DateTimeFormatter;
4
5 public class TimeZone {
6     public static void main(String[] args) {
7         LocalDateTime dateTime = LocalDateTime.of(2024, 6, 3, 12, 0);
8         ZoneId fromZone = ZoneId.of("America/New_York");
9         ZoneId toZone = ZoneId.of("Asia/Tokyo");
10        ZonedDateTime fromZonedDateTime = ZonedDateTime.of(dateTime, fromZone);
11        ZonedDateTime toZonedDateTime = fromZonedDateTime.withZoneSameInstant(toZone);
12        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
13        String fromTime = fromZonedDateTime.format(formatter);
14        String toTime = toZonedDateTime.format(formatter);
15        System.out.println("Time in " + fromZone + ": " + fromTime);
16        System.out.println("Converted time in " + toZone + ": " + toTime);
17    }
18 }
```

The console output shows the program execution:

```
<terminated> TimeZone [Java Application] C:\Users\HP\AppData\Local\Temp\wipro-workspace\Wipro\src\Assignment\TimeZone.java
America/New_York: 2024-06-03 12:00:00
Converted time in Asia/Tokyo: 2024-06-04 01:00:00
```

