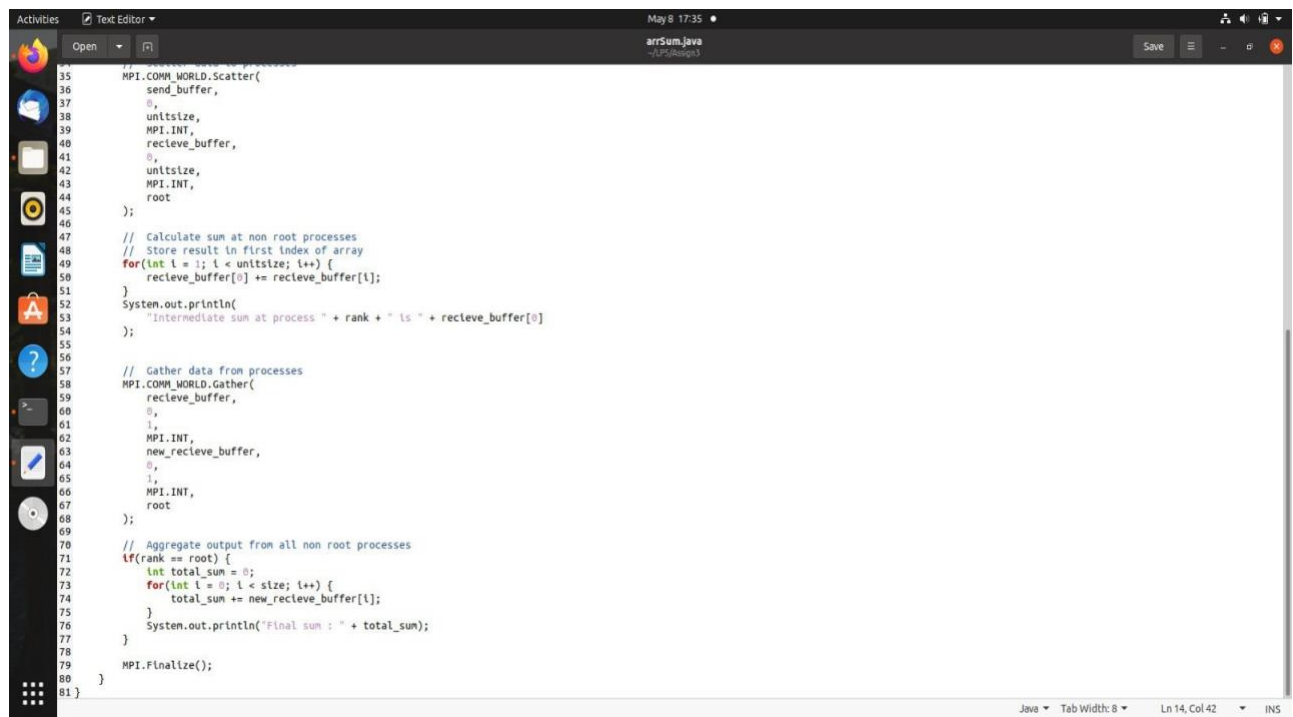


CODE IMPLEMENTATION :



This screenshot shows the first 47 lines of the `arrSum.java` file. The code includes imports for `MPI`, `Scanner`, and `Exception`. It defines a `main` method that initializes MPI, gets the rank and size, and sets up buffers. A loop for `rank == root` calculates the total number of elements and distributes data to other processes using `Scatter`. Non-root processes receive data and calculate a partial sum.

```
1 //arrSum.java
2
3 import mpi.MPI;
4
5 import java.util.Scanner;
6
7 import mpi.*;
8
9 public class arrSum {
10     public static void main(String[] args) throws Exception{
11         MPI.Init(args);
12
13         int rank = MPI.COMM_WORLD.Rank();
14         int size = MPI.COMM_WORLD.Size();
15
16         int unitSize = 5;
17         int root = 0;
18         int send_buffer[] = null;
19         // 1 process is expected to handle 4 elements
20         send_buffer = new int [unitSize * size];
21         int recieve_buffer[] = new int [unitSize];
22         int new_recieve_buffer[] = new int [size];
23
24         // Set data for distribution
25         if(rank == root) {
26             int total_elements = unitSize * size;
27             System.out.println("Enter " + total_elements + " elements");
28             for(int i = 0; i < total_elements; i++) {
29                 System.out.println("Element " + i + "\t = " + i);
30                 send_buffer[i] = i;
31             }
32         }
33
34         // Scatter data to processes
35         MPI.COMM_WORLD.Scatter(
36             send_buffer,
37             0,
38             unitSize,
39             MPI.INT,
40             recieve_buffer,
41             0,
42             unitSize,
43             MPI.INT,
44             root
45         );
46
47         // Calculate sum at non root processes
```



This screenshot shows the continuation of the `arrSum.java` code from line 48 to 81. It includes the calculation of the partial sum at non-root processes, the gathering of data back to the root process using `Gather`, and the final aggregation of the sum at the root process. The code concludes with `MPI.Finalize()`.

```
48 // Store result in first index of array
49 for(int i = 1; i < unitSize; i++) {
50     recieve_buffer[0] += recieve_buffer[i];
51 }
52 System.out.println(
53     "Intermediate sum at process " + rank + " is " + recieve_buffer[0]
54 );
55
56 // Gather data from processes
57 MPI.COMM_WORLD.Gather(
58     recieve_buffer,
59     0,
60     1,
61     MPI.INT,
62     new_recieve_buffer,
63     0,
64     1,
65     MPI.INT,
66     root
67 );
68
69 // Aggregate output from all non root processes
70 if(rank == root) {
71     int total_sum = 0;
72     for(int i = 0; i < size; i++) {
73         total_sum += new_recieve_buffer[i];
74     }
75     System.out.println("Final sum : " + total_sum);
76 }
77
78 MPI.Finalize();
79
80 }
81 }
```

Activities Text Editor May 8 17:32 Assign3.java ~/LP5/Assign3 Save

```
1 //Assign3.java
2
3 import mpi.*;
4 public class Assign3 {
5     public static void main(String args[]) throws Exception {
6         MPI.Init(args);
7         int ne = MPI.COMM_WORLD.Rank();
8         int size = MPI.COMM_WORLD.Size();
9         System.out.println("Hi from <+ne+>");
10        MPI.Finalize();
11    }
12 }
```

Java Tab Width: 8 Ln 1, Col 15 INS

Activities Terminal May 22 15:17 valbhav@valbhav-VirtualBox: ~/LP5/Assign3

```
valbhav@valbhav-VirtualBox:~/LP5/Assign3$ export MPJ_HOME=/home/valbhav/Downloads/mpj-v0.44
valbhav@valbhav-VirtualBox:~/LP5/Assign3$ export PATH=$MPJ_HOME/bin:$PATH
valbhav@valbhav-VirtualBox:~/LP5/Assign3$ javac -cp $MPJ_HOME/lib/mpj.jar arrSum.java
valbhav@valbhav-VirtualBox:~/LP5/Assign3$ $MPJ_HOME/bin/mpjrun.sh -np 4 arrSum
MPJ Express (0.44) is started in the multicore configuration
Enter 20 elements
Element 0 = 1
Element 1 = 2
Element 2 = 3
Element 3 = 4
Element 4 = 5
Element 5 = 6
Element 6 = 7
Element 7 = 8
Element 8 = 9
Element 9 = 10
Element 10 = 11
Element 11 = 12
Element 12 = 13
Element 13 = 14
Element 14 = 15
Element 15 = 16
Element 16 = 17
Element 17 = 18
Element 18 = 19
Element 19 = 20
Intermediate sum at process 3 is 90
Intermediate sum at process 2 is 65
Intermediate sum at process 1 is 40
Intermediate sum at process 0 is 15
Final sum : 210
valbhav@valbhav-VirtualBox:~/LP5/Assign3$
```