

Generics, Multithreading & GOF Design Patterns

Assignment 1

Name : Arijit Saha , College : Future Institute of Engineering and Management ,
Employee Id : T23010330 , Mail Id : arijits@trainee.nrifintech.com

Q1.

collection -> ArrayList

ask choice

CRUD-> 1. Insert -> ask emp info

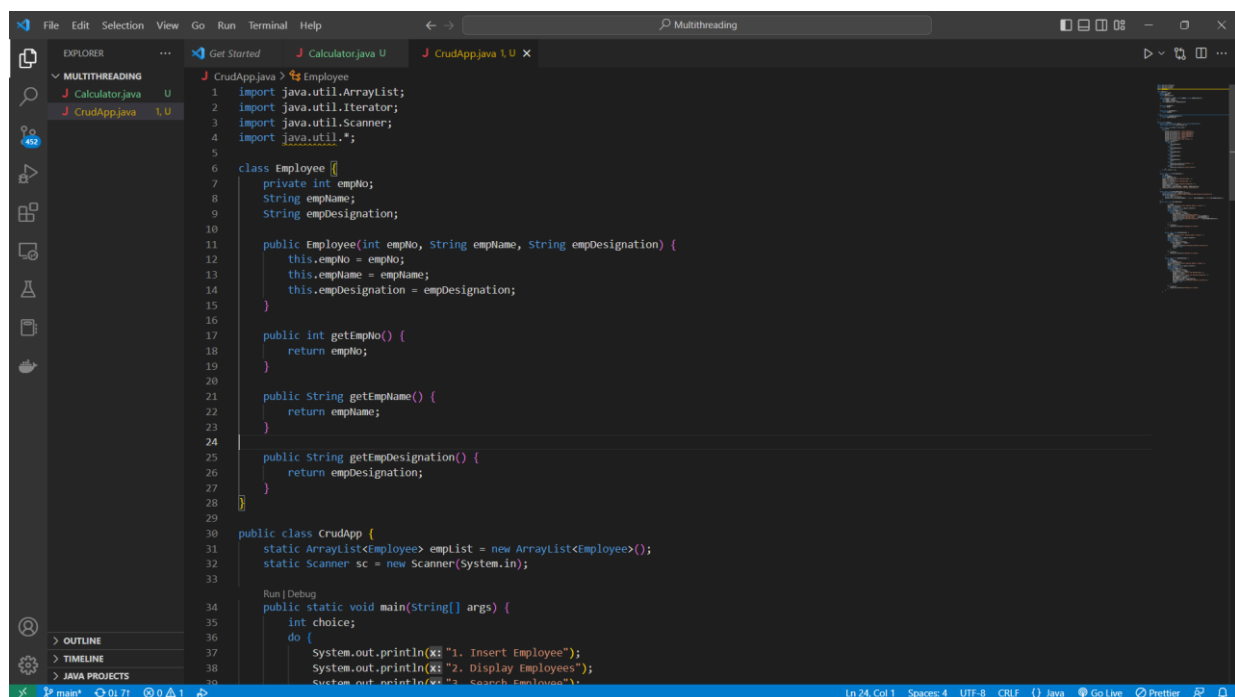
2. Display -> iterator

3. Search -> ask empno-> found else not found

4. Delete -> ask empno

5. Update -> ask empno

CrudApp.java



```
1 import java.util.ArrayList;
2 import java.util.Iterator;
3 import java.util.Scanner;
4 import java.util.*;
5
6 class Employee {
7     private int empNo;
8     String empName;
9     String empDesignation;
10
11     public Employee(int empNo, String empName, String empDesignation) {
12         this.empNo = empNo;
13         this.empName = empName;
14         this.empDesignation = empDesignation;
15     }
16
17     public int getEmpNo() {
18         return empNo;
19     }
20
21     public String getEmpName() {
22         return empName;
23     }
24
25     public String getEmpDesignation() {
26         return empDesignation;
27     }
28 }
29
30 public class CrudApp {
31     static ArrayList<Employee> empList = new ArrayList<Employee>();
32     static Scanner sc = new Scanner(System.in);
33
34     Run | Debug
35     public static void main(String[] args) {
36         int choice;
37         do {
38             System.out.println("\n1. Insert Employee");
39             System.out.println("\n2. Display Employees");
40             System.out.println("\n3. Search Employee");
41         } while (choice != 0);
42     }
43 }
```

```
File Edit Selection View Go Run Terminal Help
Multithreading

EXPLORER
MULTITHREADING
  Calculator.java U
  CrudApp.java 1.U
  Employee

J CrudApp.java > Employee
36 do {
37     System.out.println("\n 1. Insert Employee");
38     System.out.println("\n 2. Display Employees");
39     System.out.println("\n 3. Search Employee");
40     System.out.println("\n 4. Delete Employee");
41     System.out.println("\n 5. Update Employee");
42     System.out.println("\n 6. Exit");
43     System.out.print("\n Enter your choice: ");
44     choice = sc.nextInt();
45     switch (choice) {
46     case 1:
47         insertEmployee();
48         break;
49     case 2:
50         displayEmployees();
51         break;
52     case 3:
53         searchEmployee();
54         break;
55     case 4:
56         deleteEmployee();
57         break;
58     case 5:
59         updateEmployee();
60         break;
61     case 6:
62         System.out.println("\n Exiting...");
63         System.exit(status 0);
64     default:
65         System.out.println("\n Invalid choice");
66     }
67 } while (choice != 6);
68
69
70 public static void insertEmployee() {
71     int empNo;
72     String empName;
73     String empDesignation;
74     System.out.print("\n Enter Employee Number: ");
```

```
72     String empName;
73     String empDesignation;
74     System.out.print("\n Enter Employee Number: ");
75     empNo = sc.nextInt();
76     System.out.print("\n Enter Employee Name: ");
77     empName = sc.next();
78     System.out.print("\n Enter Employee Designation: ");
79     empDesignation = sc.next();
80     empList.add(new Employee(empNo, empName, empDesignation));
81     System.out.println("\n Employee Inserted successfully");
82 }
83
84 public static void displayEmployees() {
85     Iterator<Employee> itr = empList.iterator();
86     System.out.println("\n Employee Number\tEmployee Name\tEmployee Designation");
87     while (itr.hasNext()) {
88         Employee emp = itr.next();
89         System.out.println(emp.getEmpNo() + "\t\t" + emp.getEmpName() + "\t\t" + emp.getEmpDesignation());
90     }
91 }
92 public static void searchEmployee()
93 {
94     int empNo;
95     System.out.println("\n Enter Employee Number to search: ");
96     empNo = sc.nextInt();
97     Iterator<Employee> itr = empList.iterator();
98     boolean isFound = false;
99     while (itr.hasNext()) {
100         Employee emp = itr.next();
101         if (emp.getEmpNo() == empNo) {
102             System.out.println("\n Employee Found");
103             System.out.println("Employee Number: " + emp.getEmpNo());
104             System.out.println("Employee Name: " + emp.getEmpName());
105             System.out.println("Employee Designation: " + emp.getEmpDesignation());
106             isFound = true;
107             break;
108         }
109     }
110     if (!isFound) {
111         System.out.println("\n Employee not found");
112     }
113 }
```

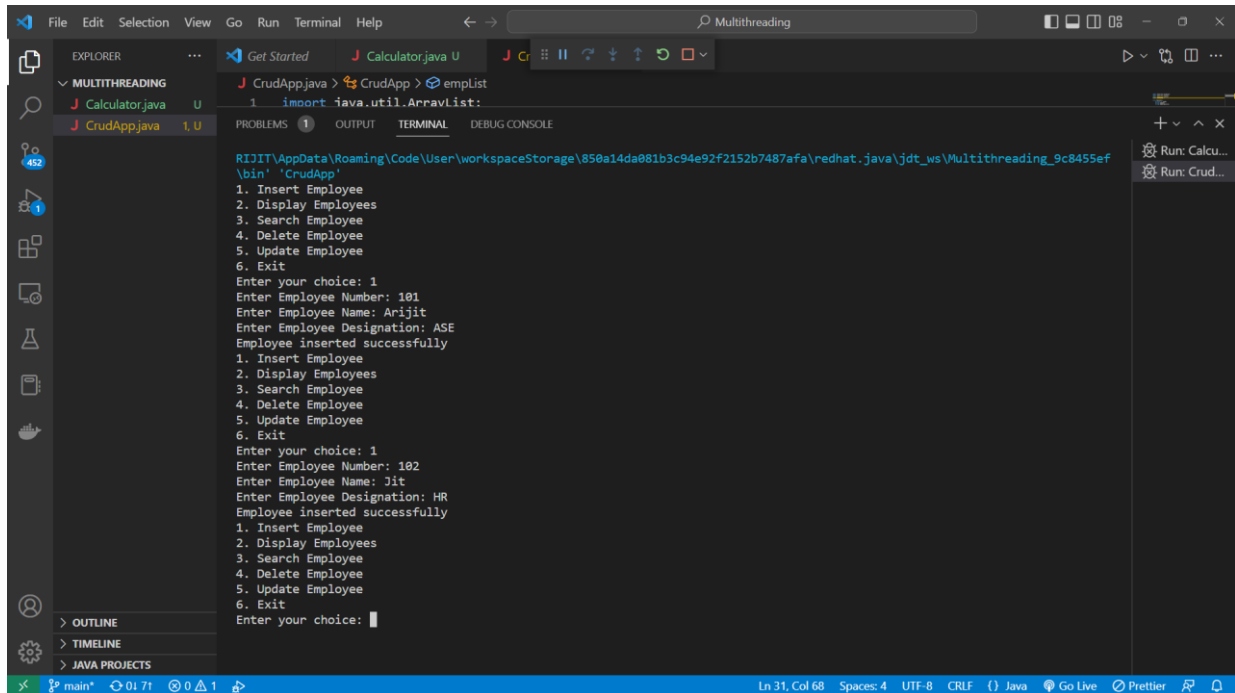
This screenshot shows the `deleteEmployee()` method in `CrudApp.java`. The method takes an integer `empNo` as input and attempts to delete an employee from a list. It uses a `while` loop with `itr.hasNext()` to iterate through the list. If an employee with the specified number is found, it is removed, and the method prints a success message. If not found, it prints a message indicating the employee was not found. The `if (!isFound)` check is placed after the loop.

```
107         break;
108     }
109 }
110 if (!isFound) {
111     System.out.println("Employee not found");
112 }
113 }
114
115
116 public static void deleteEmployee() {
117     int empNo;
118     System.out.print("Enter Employee Number to delete: ");
119     empNo = sc.nextInt();
120     Iterator<Employee> itr = empList.iterator();
121     boolean isFound = false;
122     while (itr.hasNext()) {
123         Employee emp = itr.next();
124         if (emp.getEmpNo() == empNo) {
125             itr.remove();
126             System.out.println("Employee deleted successfully");
127             isFound = true;
128             break;
129         }
130     }
131     if (!isFound) {
132         System.out.println("Employee not found");
133     }
134 }
135
136
137 public static void updateEmployee() {
138     int empNo;
139     String empName;
140     String empDesignation;
141     System.out.println("Enter Employee Number to update: ");
142     empNo = sc.nextInt();
143     Iterator<Employee> itr = empList.iterator();
144     boolean isFound = false;
145     while (itr.hasNext()) {
146         Employee emp = itr.next();
```

This screenshot shows the `updateEmployee()` method in `CrudApp.java`. The method takes an integer `empNo` and new employee details (`empName` and `empDesignation`) as input. It uses a `while` loop with `itr.hasNext()` to iterate through the list. If an employee with the specified number is found, it updates the employee's name and designation, prints a success message, and breaks the loop. If not found, it prints a message indicating the employee was not found. The `if (!isFound)` check is placed after the loop.

```
130     }
131     if (!isFound) {
132         System.out.println("Employee not found");
133     }
134 }
135
136
137 public static void updateEmployee() {
138     int empNo;
139     String empName;
140     String empDesignation;
141     System.out.println("Enter Employee Number to update: ");
142     empNo = sc.nextInt();
143     Iterator<Employee> itr = empList.iterator();
144     boolean isFound = false;
145     while (itr.hasNext()) {
146         Employee emp = itr.next();
147         if (emp.getEmpNo() == empNo) {
148             System.out.print("Enter new Employee Name: ");
149             empName = sc.next();
150             System.out.print("Enter new Employee Designation: ");
151             empDesignation = sc.next();
152             emp.empName = empName;
153             emp.empDesignation = empDesignation;
154             System.out.println("Employee updated successfully");
155             isFound = true;
156             break;
157         }
158     }
159     if (!isFound) {
160         System.out.println("Employee not found");
161     }
162 }
163
164 }
```

Output:

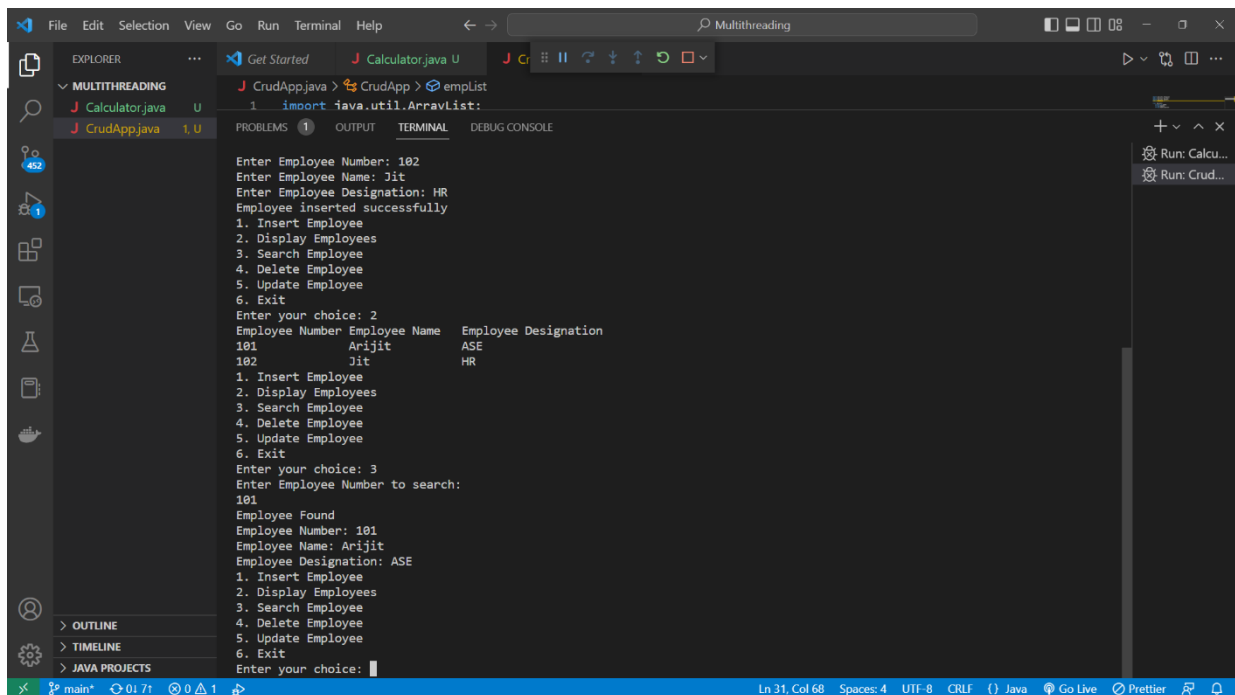


```
File Edit Selection View Go Run Terminal Help
Multithreading

EXPLORER
  MULTITHREADING
    Calculator.java U
    CrudApp.java 1, U

J CrudApp.java > CrudApp > emplist
1 import java.util.ArrayList:

R:\JIT\AppData\Roaming\Code\User\workspaceStorage\850e14da081b3c94e92f2152b7487afa\redhat.java\jdt_ws\Multithreading_9c8455ef\bin\ 'CrudApp'
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 1
Enter Employee Number: 101
Enter Employee Name: Arijit
Enter Employee Designation: ASE
Employee inserted successfully
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 1
Enter Employee Number: 102
Enter Employee Name: Jit
Enter Employee Designation: HR
Employee inserted successfully
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 
```

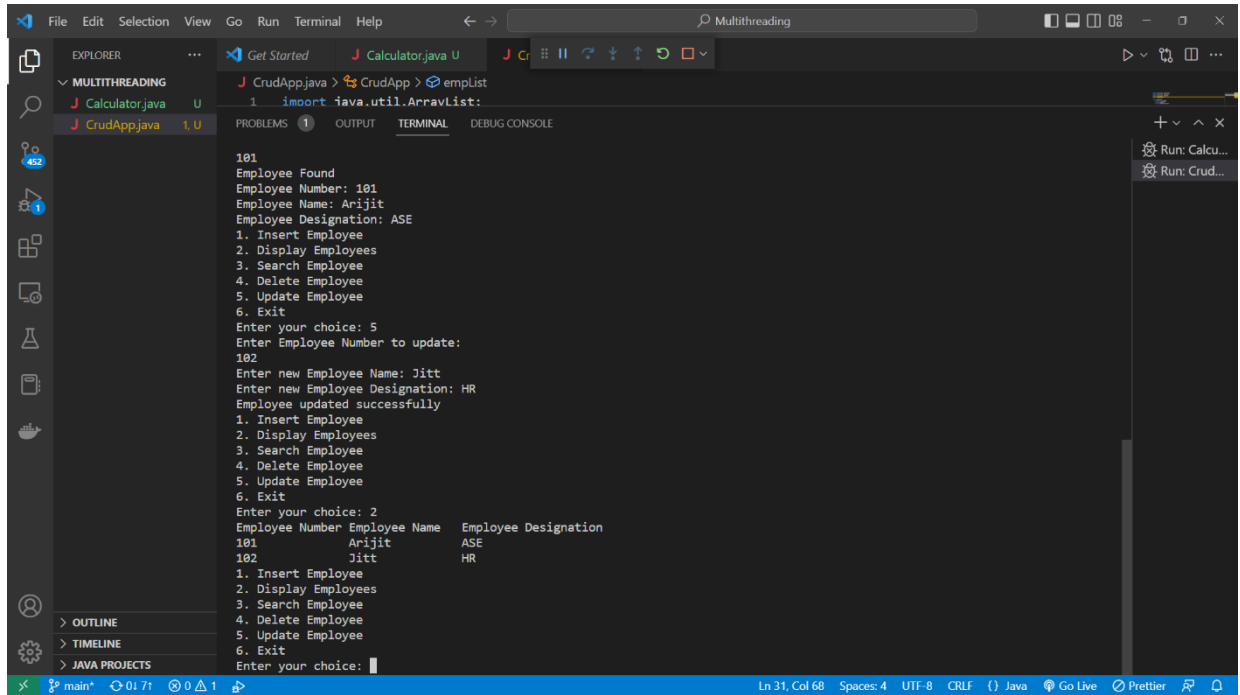


```
File Edit Selection View Go Run Terminal Help
Multithreading

EXPLORER
  MULTITHREADING
    Calculator.java U
    CrudApp.java 1, U

J CrudApp.java > CrudApp > emplist
1 import java.util.ArrayList:

Enter Employee Number: 102
Enter Employee Name: Jit
Enter Employee Designation: HR
Employee inserted successfully
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 2
Employee Number Employee Name Employee Designation
101 Arijit ASE
102 Jit HR
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 3
Enter Employee Number to search:
101
Employee Found
Employee Number: 101
Employee Name: Arijit
Employee Designation: ASE
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 
```



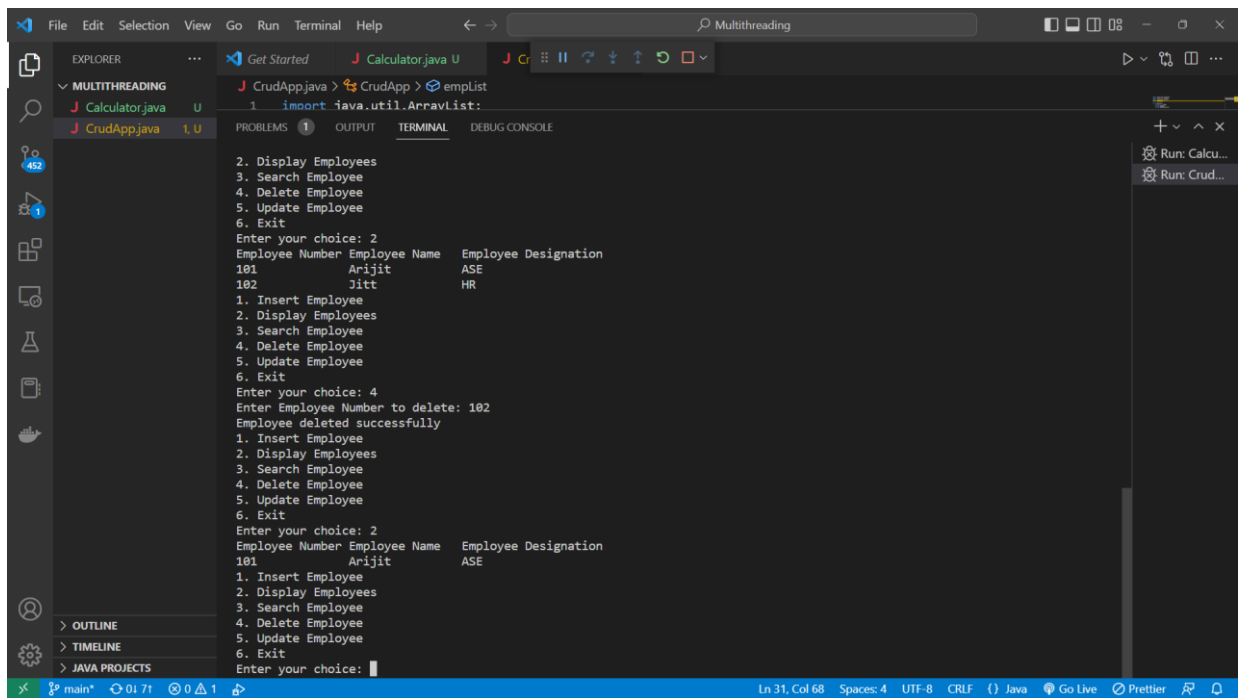
```
File Edit Selection View Go Run Terminal Help
Multithreading

EXPLORER
MULTITHREADING
  Calculator.java U
  CrudApp.java 1, U

J CrudApp.java > CrudApp > emplList
1 import java.util.ArrayList;

TERMINAL
101
Employee Found
Employee Number: 101
Employee Name: Arijit
Employee Designation: ASE
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 5
Enter Employee Number to update:
102
Enter new Employee Name: Jitt
Enter new Employee Designation: HR
Employee updated successfully
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 2
Employee Number Employee Name Employee Designation
101 Arijit ASE
102 Jitt HR
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice:

main* 0:71 0 1 Ln 31, Col 68 Spaces: 4 UTF-8 CRLF () Java Go Live Prettier
```



```
File Edit Selection View Go Run Terminal Help
Multithreading

EXPLORER
MULTITHREADING
  Calculator.java U
  CrudApp.java 1, U

J CrudApp.java > CrudApp > emplList
1 import java.util.ArrayList;

TERMINAL
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 2
Employee Number Employee Name Employee Designation
101 Arijit ASE
102 Jitt HR
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 4
Enter Employee Number to delete: 102
Employee deleted successfully
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice: 2
Employee Number Employee Name Employee Designation
101 Arijit ASE
1. Insert Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Exit
Enter your choice:

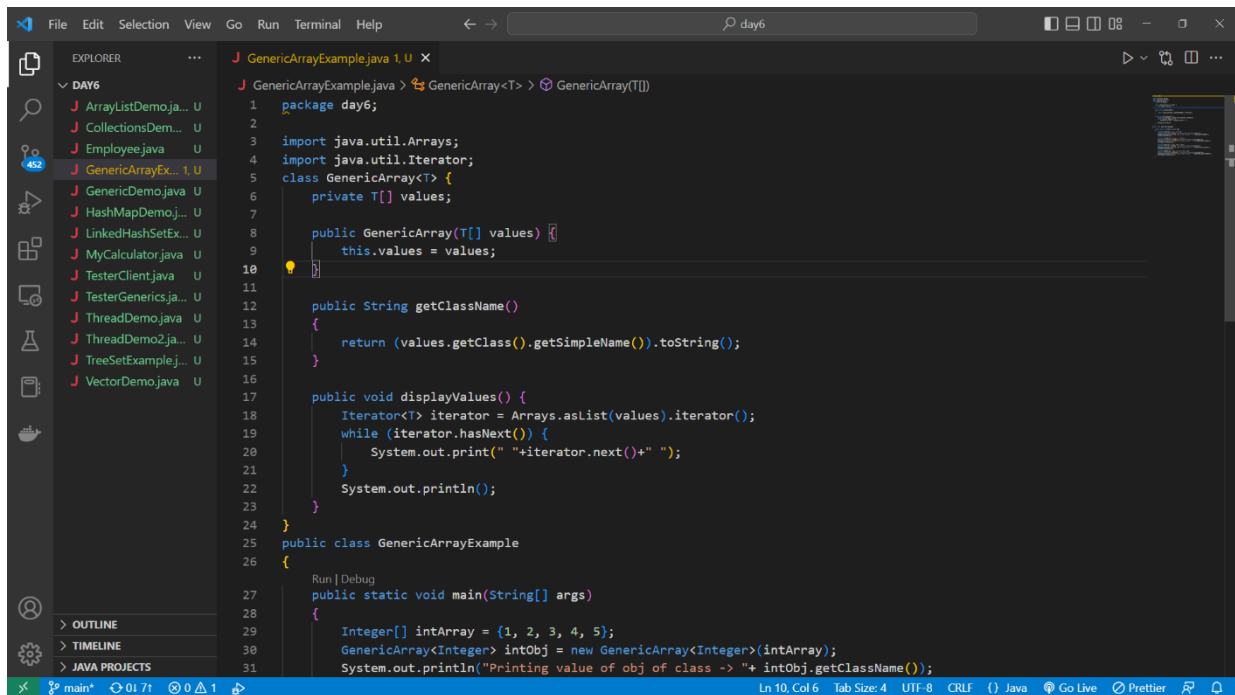
main* 0:71 0 1 Ln 31, Col 68 Spaces: 4 UTF-8 CRLF () Java Go Live Prettier
```

Q2. Class Generic type

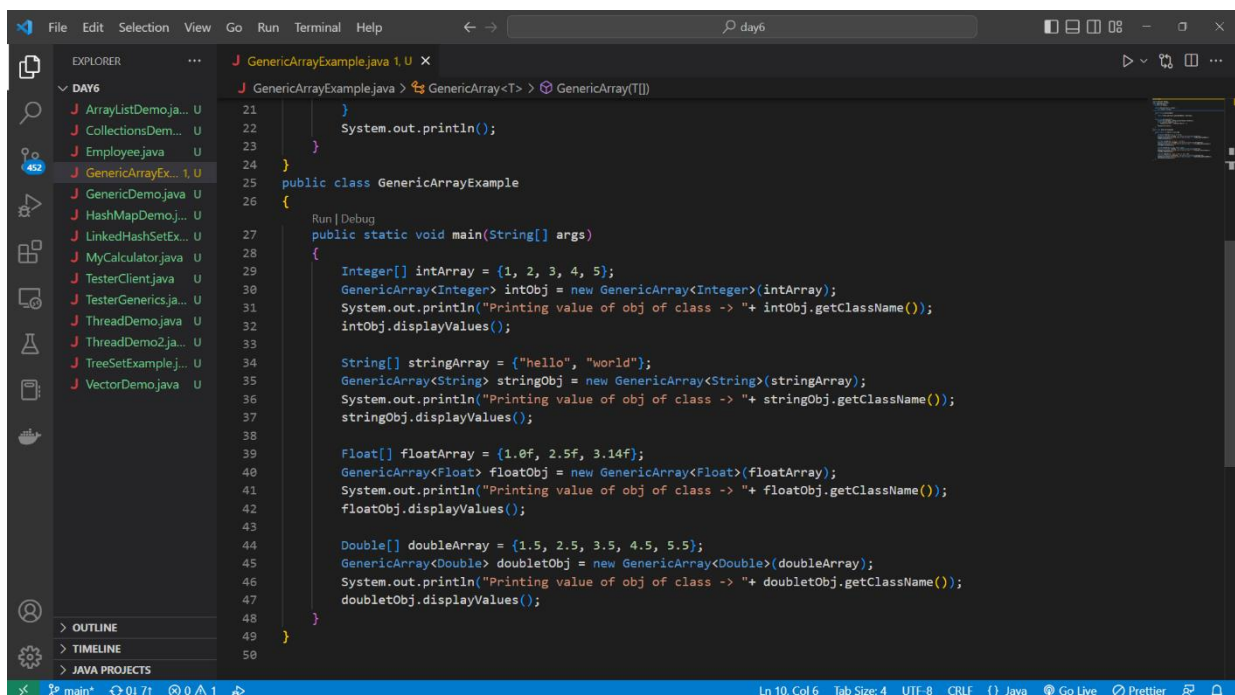
method -> print/display -> values of array -> iterator

integer, float, string, double

GenericArrayExample.java

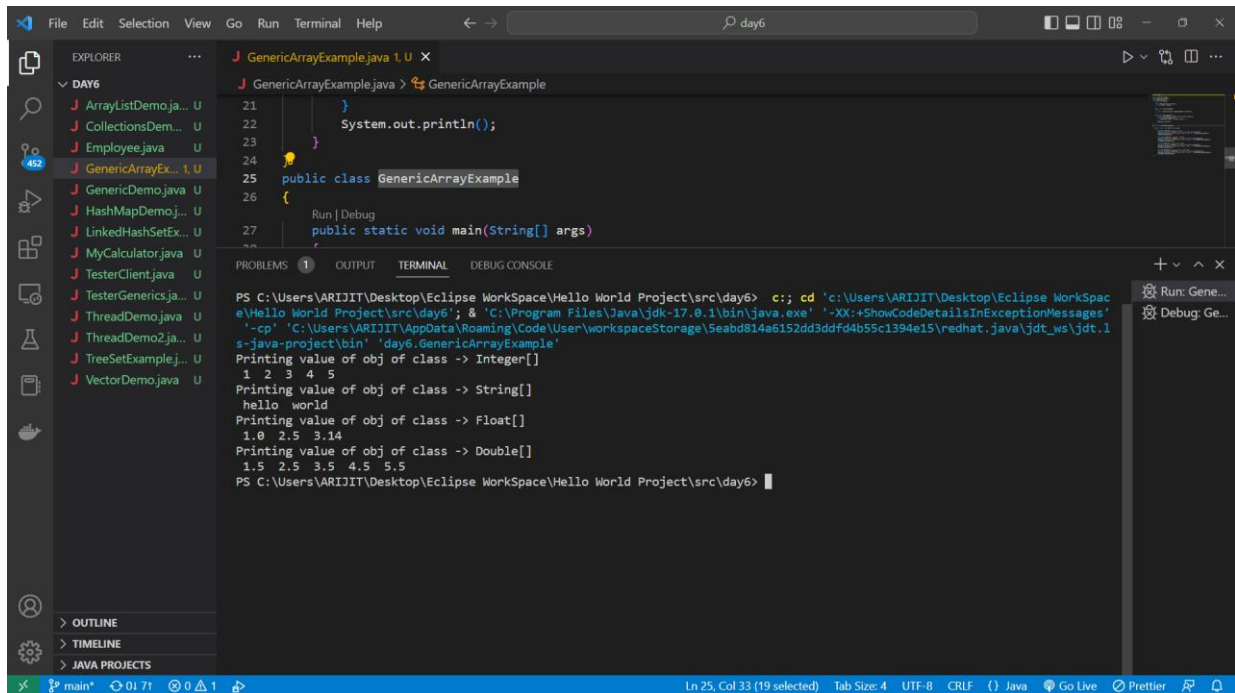


```
1 package day6;
2
3 import java.util.Arrays;
4 import java.util.Iterator;
5 class GenericArray<T> {
6     private T[] values;
7
8     public GenericArray(T[] values) {
9         this.values = values;
10    }
11
12    public String getClassNames()
13    {
14        return (values.getClass().getSimpleName()).toString();
15    }
16
17    public void displayValues() {
18        Iterator<T> iterator = Arrays.asList(values).iterator();
19        while (iterator.hasNext()) {
20            System.out.print(" "+iterator.next()+" ");
21        }
22        System.out.println();
23    }
24 }
25 public class GenericArrayExample
26 {
27     Run | Debug
28     public static void main(String[] args)
29     {
30         Integer[] intArray = {1, 2, 3, 4, 5};
31         GenericArray<Integer> intObj = new GenericArray<Integer>(intArray);
32         System.out.println("Printing value of obj of class -> "+ intObj.getClassNames());
```



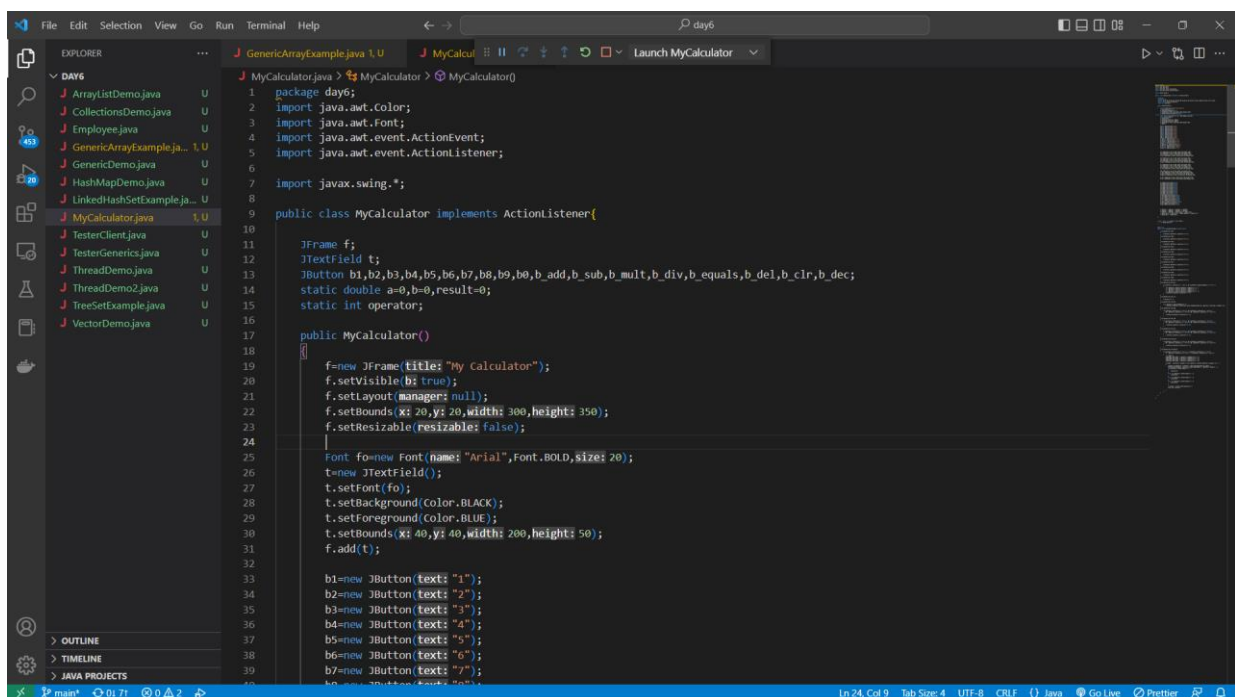
```
21    }
22    System.out.println();
23    }
24 }
25 public class GenericArrayExample
26 {
27     Run | Debug
28     public static void main(String[] args)
29     {
30         Integer[] intArray = {1, 2, 3, 4, 5};
31         GenericArray<Integer> intObj = new GenericArray<Integer>(intArray);
32         System.out.println("Printing value of obj of class -> "+ intObj.getClassNames());
33         intObj.displayValues();
34
35         String[] stringArray = {"hello", "world"};
36         GenericArray<String> stringObj = new GenericArray<String>(stringArray);
37         System.out.println("Printing value of obj of class -> "+ stringObj.getClassNames());
38         stringObj.displayValues();
39
40         Float[] floatArray = {1.0f, 2.5f, 3.14f};
41         GenericArray<Float> floatObj = new GenericArray<Float>(floatArray);
42         System.out.println("Printing value of obj of class -> "+ floatObj.getClassNames());
43         floatObj.displayValues();
44
45         Double[] doubleArray = {1.5, 2.5, 3.5, 4.5, 5.5};
46         GenericArray<Double> doubleObj = new GenericArray<Double>(doubleArray);
47         System.out.println("Printing value of obj of class -> "+ doubleObj.getClassNames());
48         doubleObj.displayValues();
49     }
50 }
```

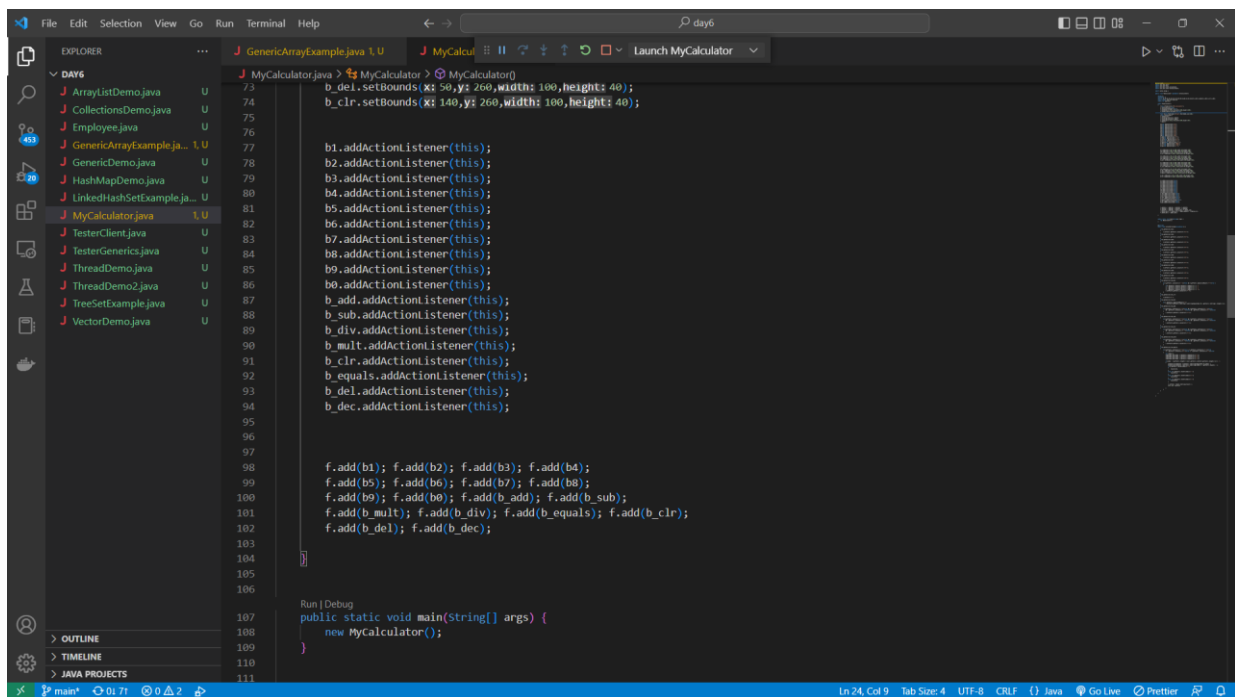
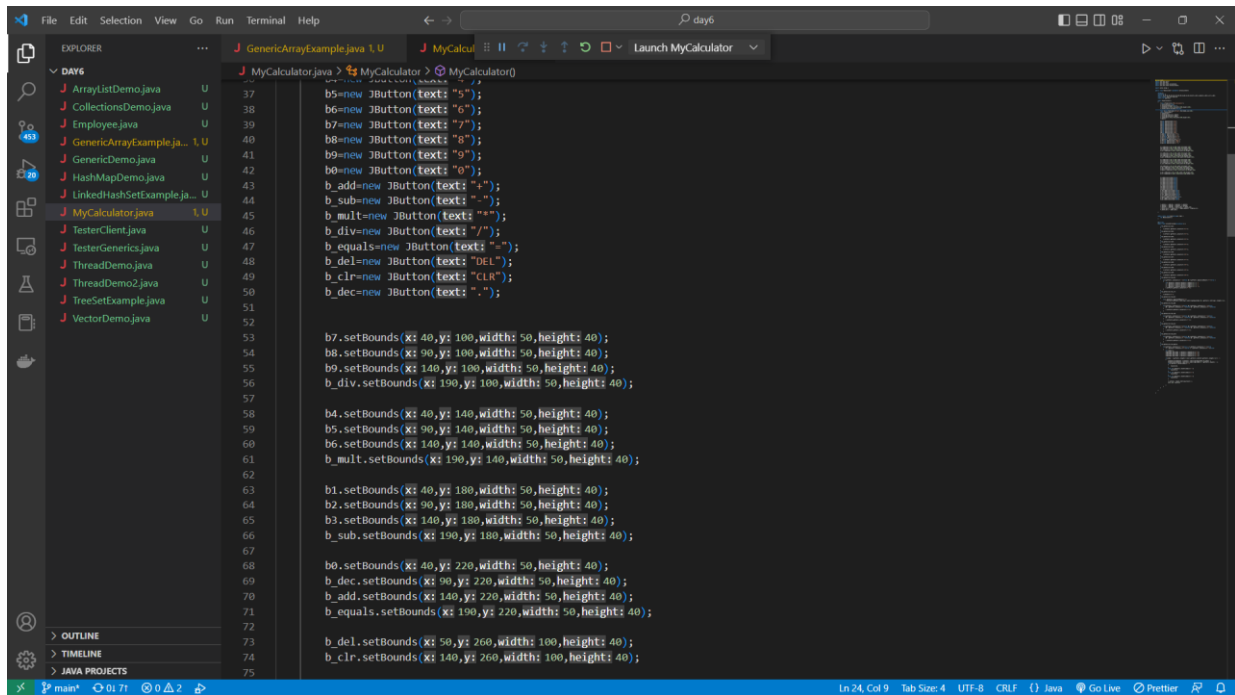
Output:



Q3. Complete GUI Calculator using swing

MyCalculator.java





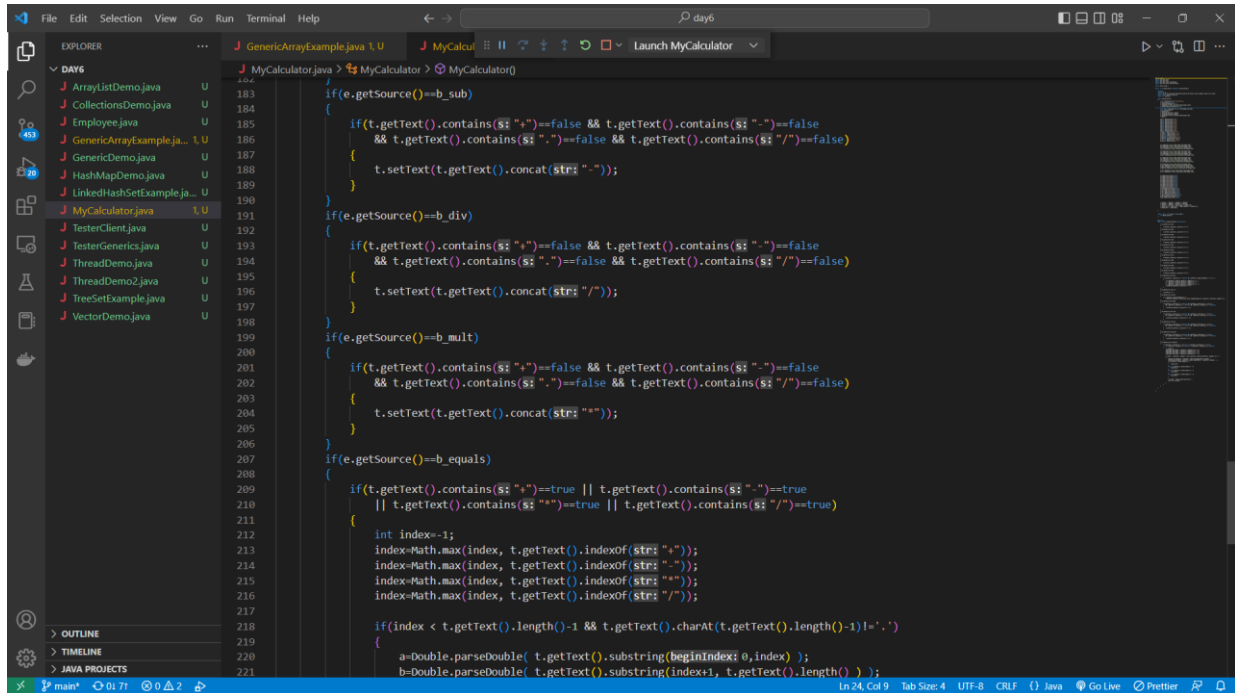

```
@Override
public void actionPerformed(ActionEvent e) {

    if(e.getSource()==b1)
    {
        t.setText(t.getText().concat(String.valueOf(1)));
    }
    if(e.getSource()==b2)
    {
        t.setText(t.getText().concat(String.valueOf(2)));
    }
    if(e.getSource()==b3)
    {
        t.setText(t.getText().concat(String.valueOf(3)));
    }
    if(e.getSource()==b4)
    {
        t.setText(t.getText().concat(String.valueOf(4)));
    }
    if(e.getSource()==b5)
    {
        t.setText(t.getText().concat(String.valueOf(5)));
    }
    if(e.getSource()==b6)
    {
        t.setText(t.getText().concat(String.valueOf(6)));
    }
    if(e.getSource()==b7)
    {
        t.setText(t.getText().concat(String.valueOf(7)));
    }
    if(e.getSource()==b8)
    {
        t.setText(t.getText().concat(String.valueOf(8)));
    }
    if(e.getSource()==b9)
    {
        t.setText(t.getText().concat(String.valueOf(9)));
    }
}
```

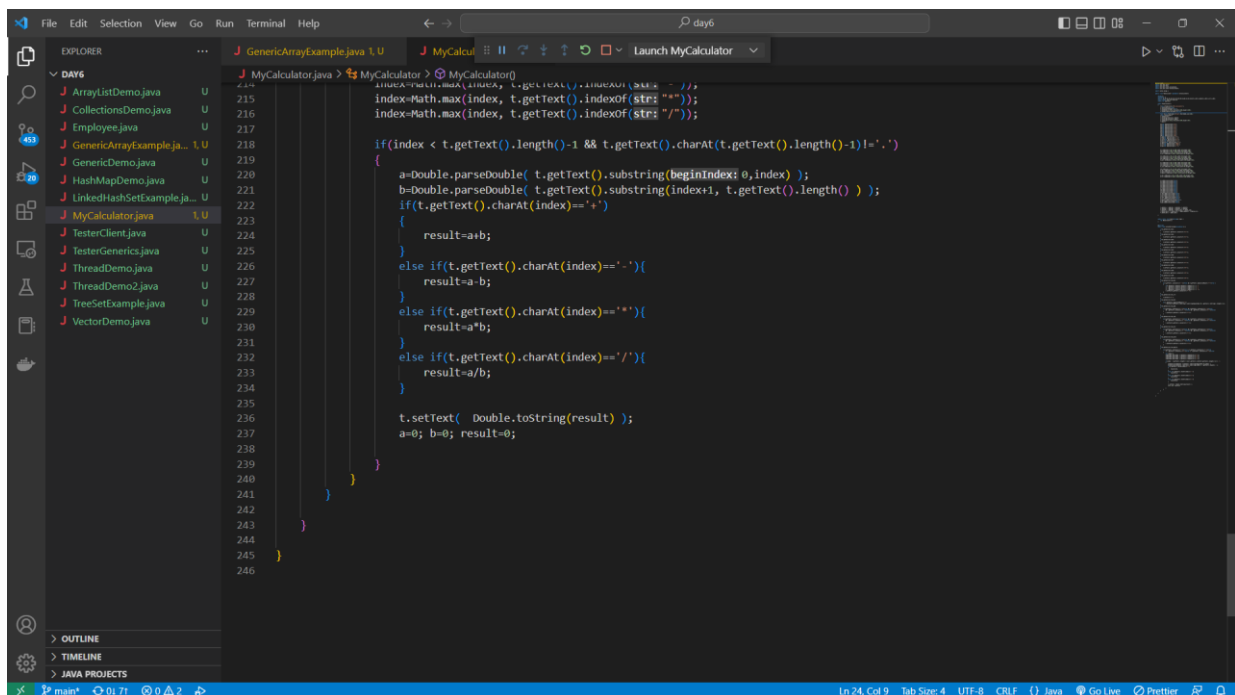
```

    if(e.getSource()==b_dec)
    {
        if(t.getText().contains(".")==false && t.getText().equals("")!=true )
        {
            if(t.getText().charAt(t.getText().length()-1) != '+'
            || t.getText().charAt(t.getText().length()-1) != '-'
            || t.getText().charAt(t.getText().length()-1) != '*'
            || t.getText().charAt(t.getText().length()-1) != '/')
            t.setText(t.getText().concat(String.valueOf(".")));
        }
    }
    if(e.getSource()==b_clr)
    {
        t.setText("");
    }
    if(e.getSource()==b_del)
    {
        if(!t.getText().equals(""))
        t.setText(t.getText().toString().substring(beginIndex: 0,t.getText().toString().length()-1));
    }
    if(e.getSource()==b_add)
    {
        if(t.getText().contains("+")==false && t.getText().contains("-")==false
        && t.getText().contains(".")==false && t.getText().contains("/")==false)
        {
            t.setText(t.getText().concat(String.valueOf("+")));
        }
    }
    if(e.getSource()==b_sub)
    {
        if(t.getText().contains("+")==false && t.getText().contains("-")==false

```



```
183 if(e.getSource()==b_sub)
184 {
185     if(t.getText().contains("${ "+"})==false && t.getText().contains("${ "-"})==false
186         && t.getText().contains("${ "."})==false && t.getText().contains("${ "/"})==false)
187     {
188         t.setText(t.getText().concat(str: "-"));
189     }
190 }
191 if(e.getSource()==b_div)
192 {
193     if(t.getText().contains("${ "+"})==false && t.getText().contains("${ "-"})==false
194         && t.getText().contains("${ "."})==false && t.getText().contains("${ "/"})==false)
195     {
196         t.setText(t.getText().concat(str: "/"));
197     }
198 }
199 if(e.getSource()==b_mult)
200 {
201     if(t.getText().contains("${ "+"})==false && t.getText().contains("${ "-"})==false
202         && t.getText().contains("${ "."})==false && t.getText().contains("${ "/"})==false)
203     {
204         t.setText(t.getText().concat(str: "*"));
205     }
206 }
207 if(e.getSource()==b_equals)
208 {
209     if(t.getText().contains("${ "+"})==true || t.getText().contains("${ "-"})==true
210        || t.getText().contains("${ "."})==true || t.getText().contains("${ "/"})==true)
211     {
212         int index=-1;
213         index=Math.max(index, t.getText().indexOf(str: "+"));
214         index=Math.max(index, t.getText().indexOf(str: "-"));
215         index=Math.max(index, t.getText().indexOf(str: "*"));
216         index=Math.max(index, t.getText().indexOf(str: "/"));
217
218         if(index < t.getText().length()-1 && t.getText().charAt(t.getText().length()-1)!='.')
219         {
220             a=Double.parseDouble( t.getText().substring(beginIndex, 0,index) );
221             b=Double.parseDouble( t.getText().substring(index+1, t.getText().length() ) );
```



```
222         {
223             result=a+b;
224         }
225         else if(t.getText().charAt(index)=='-'){
226             result=a-b;
227         }
228         else if(t.getText().charAt(index)=='*'){
229             result=a*b;
230         }
231         else if(t.getText().charAt(index)=='/'){
232             result=a/b;
233         }
234     }
235     t.setText( Double.toString(result) );
236     a=0; b=0; result=0;
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
```

Output :

