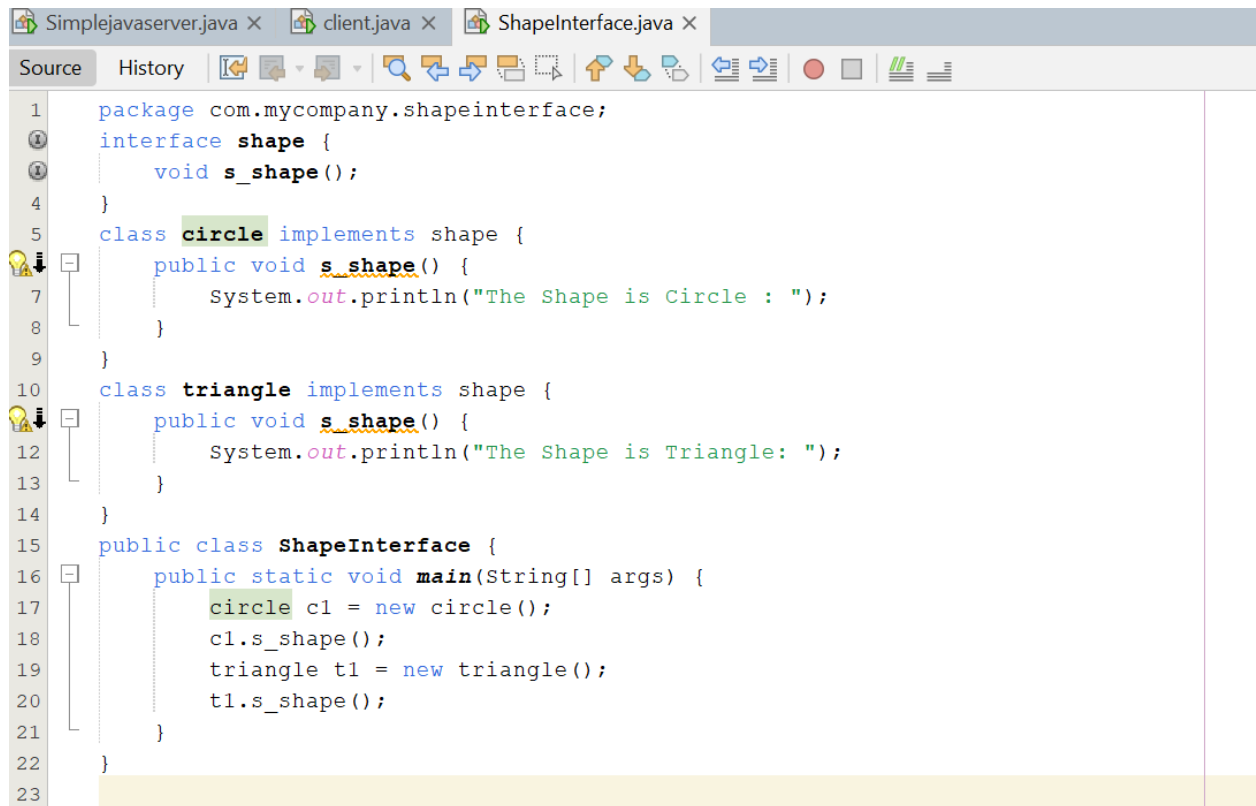


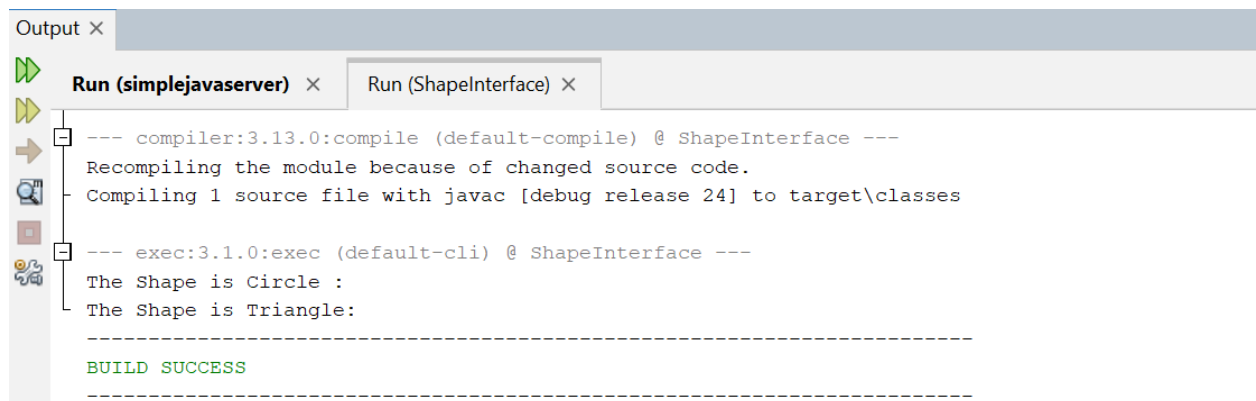
## 1. Implementing Shape Interface in Java.

### Code:



```
1 package com.mycompany.shapeinterface;
2 interface shape {
3     void s_shape();
4 }
5 class circle implements shape {
6     public void s_shape() {
7         System.out.println("The Shape is Circle : ");
8     }
9 }
10 class triangle implements shape {
11     public void s_shape() {
12         System.out.println("The Shape is Triangle: ");
13     }
14 }
15 public class ShapeInterface {
16     public static void main(String[] args) {
17         circle c1 = new circle();
18         c1.s_shape();
19         triangle t1 = new triangle();
20         t1.s_shape();
21     }
22 }
23
```

### Output:



```
Output X
Run (simplejavaserver) X Run (ShapeInterface) X
--- compiler:3.13.0:compile (default-compile) @ ShapeInterface ---
Recompiling the module because of changed source code.
Compiling 1 source file with javac [debug release 24] to target\classes
--- exec:3.1.0:exec (default-cli) @ ShapeInterface ---
The Shape is Circle :
The Shape is Triangle:
-----
BUILD SUCCESS
-----
```

## 2. Java Collections Example Using List, Set, Queue, and Map:

### Code:

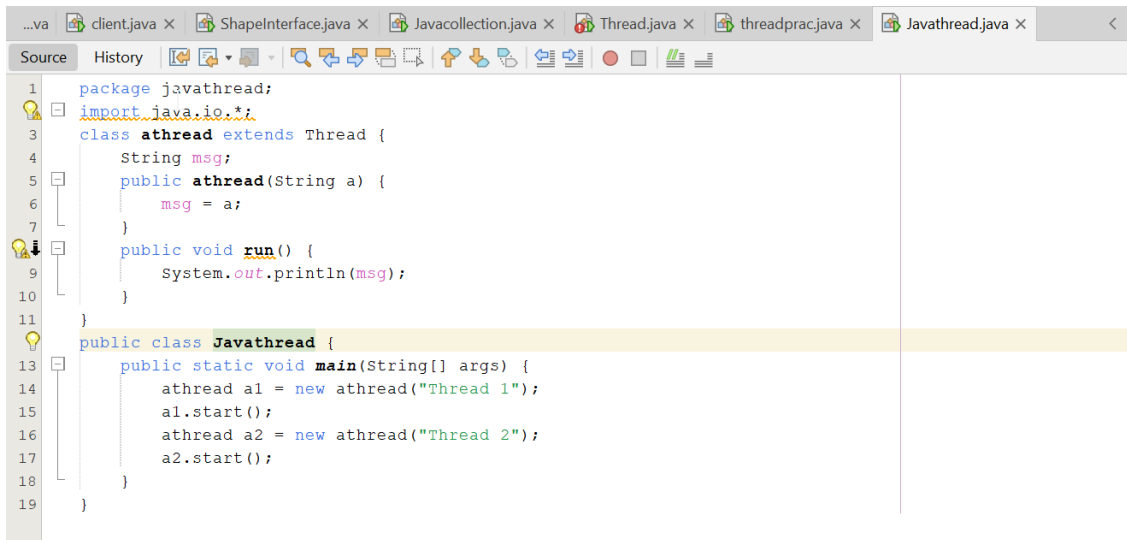
```
Simplejavaserver.java × client.java × ShapeInterface.java × Javacollection.java ×
Source History
1 package com.mycompany.javacollection;
2 import java.util.*;
3 public class Javacollection {
4     public static void main(String[] args) {
5         //list
6         List<String> book = new ArrayList<>();
7         book.add("Java Basic Knowledge");
8         book.add("Advanced Java");
9         System.out.println("Book Name: " + book);
10        System.out.println(book.size());
11        book.remove("Advanced Java");
12        System.out.println("Books Name After Removing: " + book);
13        book.clear();
14        System.out.println("After Clearing Name: " + book);
15
16        //set
17        Set<String> author = new HashSet<>();
18        author.add("Muhammad");
19        author.add("Mostakim");
20        System.out.println("Author Name: " + author);
21        author.remove("Muhammad");
22        System.out.println("After Removing Author Name: " + author);
23
24        //queue
25        Queue<String> request = new LinkedList<>();
26        request.add("Searching Books 1");
27        request.add("Searching Books 2");
28        request.add("Searching Books 3");
29        System.out.println("After Removing Searching: " + request);
30        request.clear();
31        System.out.println("After Clear: " + request);
32        System.out.println(request.size());
33
34        //map
35        Map<Integer, String> bookId = new HashMap<>();
36        bookId.put(101, "Java");
37        bookId.put(102, "Java");
38        bookId.put(103, "Java");
39        System.out.println(bookId);
40    }
```

### Output:

```
Output ×
Run (simplejavaserver) × Run (javacollection) ×
--- exec:3.1.0:exec (default-cli) @ javacollection ---
Book Name: [Java Basic Knowledge, Advanced Java]
2
Books Name After Removing: [Java Basic Knowledge]
After Clearing Name: []
Author Name: [Muhammad, Mostakim]
After Removing Author Name: [Mostakim]
After Removing Searching: [Searching Books 1, Searching Books 2, Searching Books 3]
After Clear: []
0
{101=Java, 102=Java, 103=Java}
-----
BUILD SUCCESS
-----
```

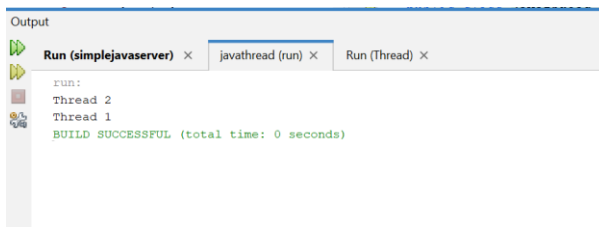
### 3. Java Thread Creation:

#### Code:



```
1 package javathread;
2 import java.io.*;
3 class athread extends Thread {
4     String msg;
5     public athread(String a) {
6         msg = a;
7     }
8     public void run() {
9         System.out.println(msg);
10    }
11 }
12
13 public class Javathread {
14     public static void main(String[] args) {
15         athread a1 = new athread("Thread 1");
16         a1.start();
17         athread a2 = new athread("Thread 2");
18         a2.start();
19     }
20 }
```

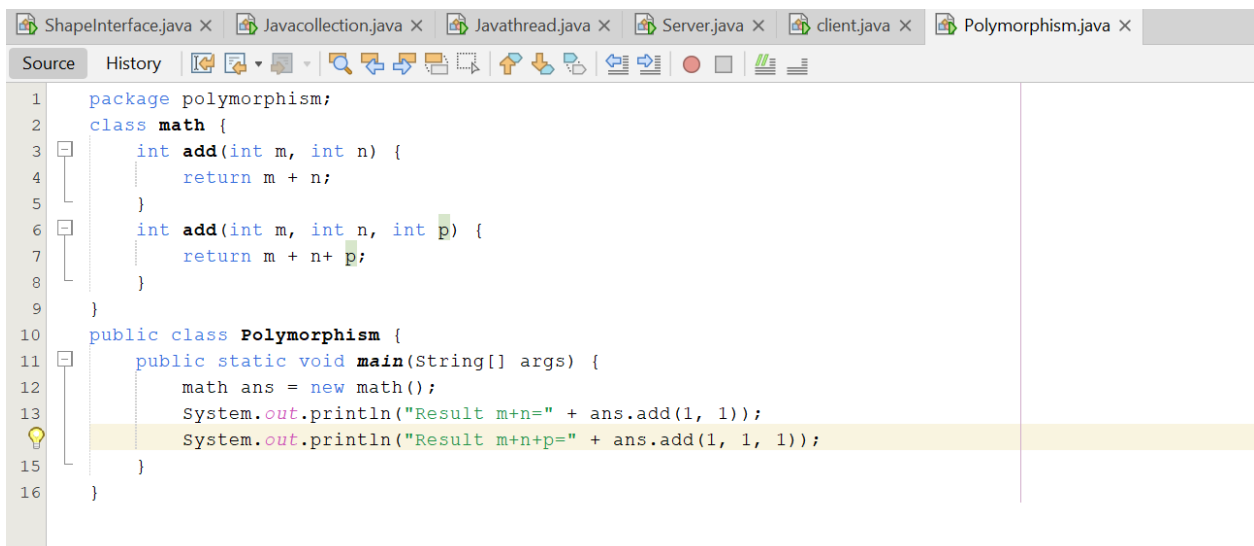
#### Output:



```
Output
Run (simplejavaserver) x javathread (run) x Run (Thread) x
run:
Thread 2
Thread 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

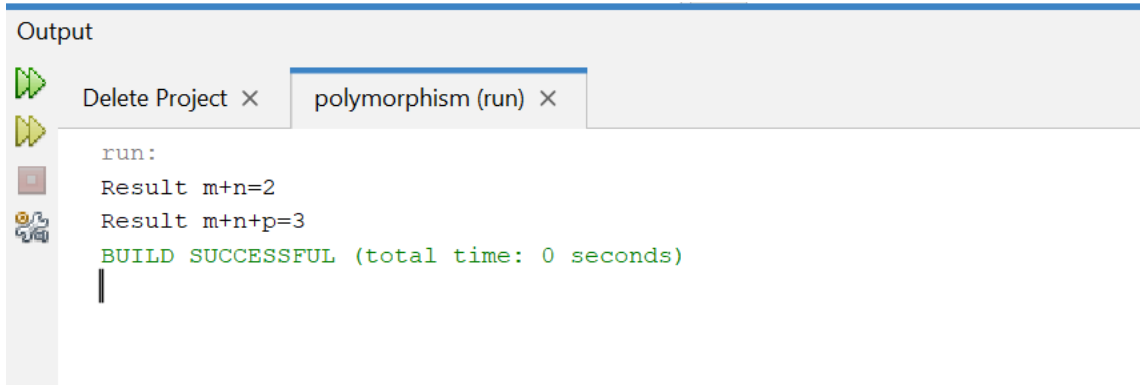
### 4. Polymorphism with Method Overloading

#### Code:



```
1 package polymorphism;
2 class math {
3     int add(int m, int n) {
4         return m + n;
5     }
6     int add(int m, int n, int p) {
7         return m + n + p;
8     }
9 }
10 public class Polymorphism {
11     public static void main(String[] args) {
12         math ans = new math();
13         System.out.println("Result m+n=" + ans.add(1, 1));
14         System.out.println("Result m+n+p=" + ans.add(1, 1, 1));
15     }
16 }
```

## Output:

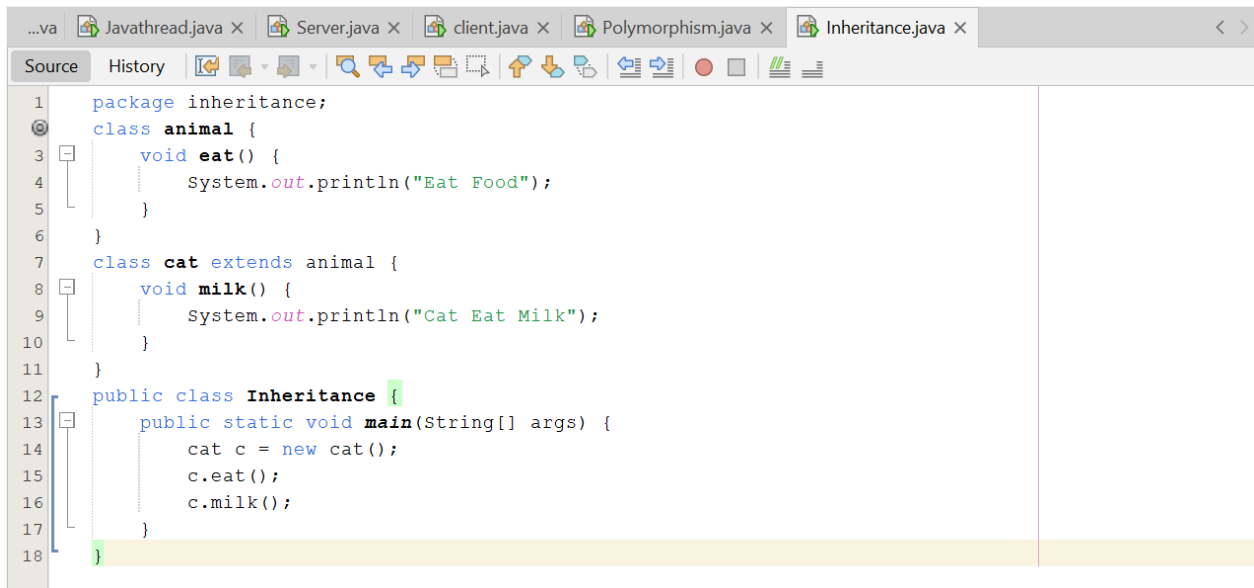


The screenshot shows an IDE output window with a tab labeled "polymorphism (run)". The output text is as follows:

```
run:
Result m+n=2
Result m+n+p=3
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 5. Inheritance in Java

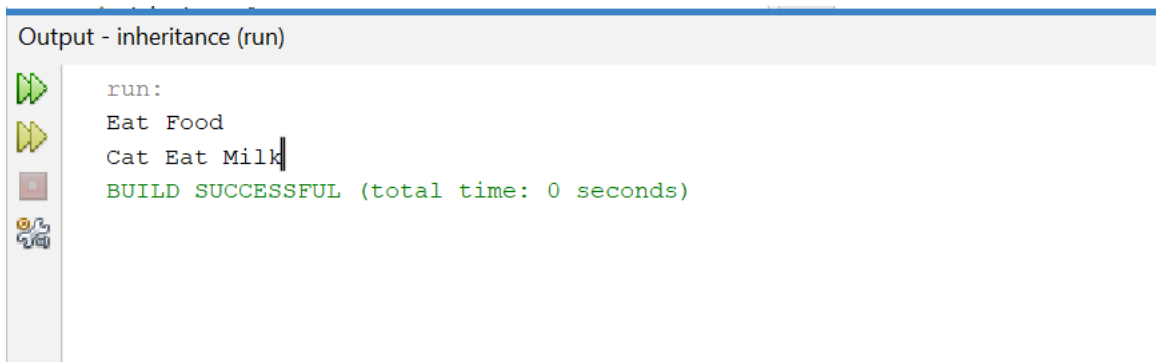
### Code:



The screenshot shows an IDE with several tabs open, including "Inheritance.java". The code in the "Inheritance.java" tab is as follows:

```
1 package inheritance;
2 class animal {
3     void eat() {
4         System.out.println("Eat Food");
5     }
6 }
7 class cat extends animal {
8     void milk() {
9         System.out.println("Cat Eat Milk");
10    }
11 }
12 public class Inheritance {
13     public static void main(String[] args) {
14         cat c = new cat();
15         c.eat();
16         c.milk();
17     }
18 }
```

## Output:

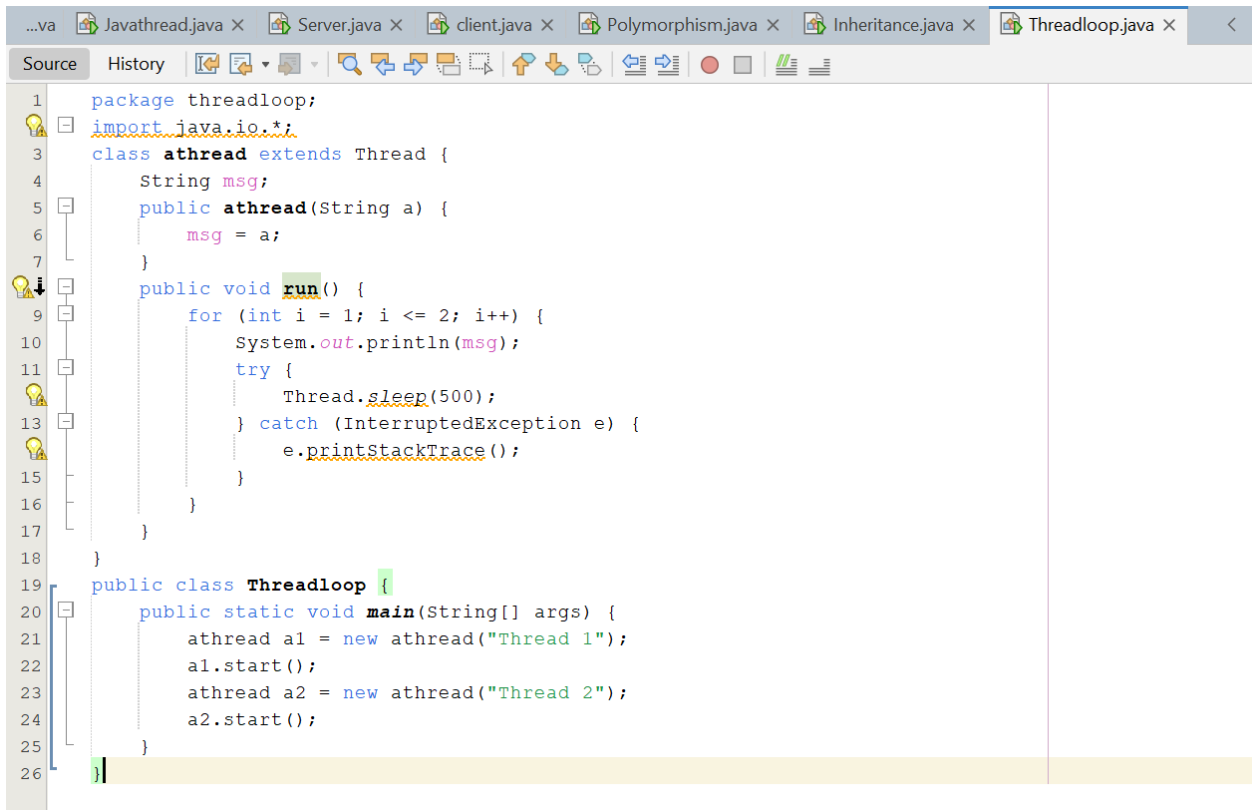


The screenshot shows an IDE output window with a tab labeled "Output - inheritance (run)". The output text is as follows:

```
run:
Eat Food
Cat Eat Milk
BUILD SUCCESSFUL (total time: 0 seconds)
```

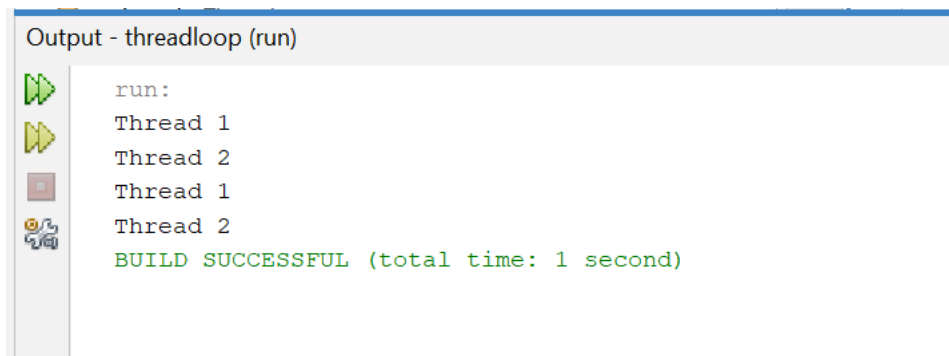
## 6. Multithreading with Loop in Java

### Code:



```
1 package threadloop;
2 import java.io.*;
3 class athread extends Thread {
4     String msg;
5     public athread(String a) {
6         msg = a;
7     }
8     public void run() {
9         for (int i = 1; i <= 2; i++) {
10             System.out.println(msg);
11             try {
12                 Thread.sleep(500);
13             } catch (InterruptedException e) {
14                 e.printStackTrace();
15             }
16         }
17     }
18 }
19 public class Threadloop {
20     public static void main(String[] args) {
21         athread a1 = new athread("Thread 1");
22         a1.start();
23         athread a2 = new athread("Thread 2");
24         a2.start();
25     }
26 }
```

### Output:



```
Output - threadloop (run)

run:
Thread 1
Thread 2
Thread 1
Thread 2
BUILD SUCCESSFUL (total time: 1 second)
```

## 7. Serialization in Java

### Code:

```
...va  Server.java x  client.java x  Polymorphism.java x  Inheritance.java x  Threadloop.java x  Serialization.java x  <
Source  History  [Icons]
1  package serialization;
2  import java.io.*;
3  class person implements Serializable {
4      public static final long serialVersionUID = 1L;
5      String name;
6      int id;
7      transient String password;
8      public person(String s, int i, String p) {
9          name = s;
10         id = i;
11         password = p;
12         System.out.println("Default Constructor");
13     }
14     public void display() {
15         System.out.println(name + "\n" + id + "\n" + password);
16     }
17 }
18 public class Serialization {
19     public static void main(String[] args) {
20         person p1 = new person("Mostakim", 1051, "PASSWORD NAI");
21         p1.display();
22         try {
23             FileOutputStream fo = new FileOutputStream("person.ser");
24             ObjectOutputStream os = new ObjectOutputStream(fo);
25             os.writeObject(p1);
26             os.close();
27             fo.close();
28             System.out.println("Object is Serialized");
29             FileInputStream fi = new FileInputStream("Person.ser");
30             ObjectInputStream os2 = new ObjectInputStream(fi);
31             person dp = (person) os2.readObject();
32             os2.close();
33             fi.close();
34             System.out.println("Object is Deserialized");
35             dp.display();
36         } catch (Exception e) {
37             System.out.println("An Error Occurred");
38             e.printStackTrace();
39         }
40     }
}
```

### Output:

```
Output - serialization (run)
run:
Default Constructor
Mostakim
1051
PASSWORD NAI
Object is Serialized
Object is Deserialized
Mostakim
1051
null
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 8. Java File Read and Write Operation

### Code:

```
1 package readwriteop;
2 import java.io.*;
3 public class Readwriteop {
4     public static void main(String[] args) {
5         try {
6             File f1 = new File("Mostakim.txt");
7             if (f1.createNewFile()) {
8                 System.out.println("Create File: " + f1.getName());
9                 System.out.println("Location: " + f1.getAbsolutePath());
10            } else {
11                System.out.println("File Already Created");
12                System.out.println("Location: " + f1.getAbsolutePath());
13            }
14            System.out.println("File Size: " + f1.length());
15            FileWriter w = new FileWriter(f1, true);
16            w.write("CITY UNIVERSITY");
17            System.out.println("Check The File");
18            w.close();
19            BufferedReader b1 = new BufferedReader(new FileReader(f1));
20            String x;
21            while ((x = b1.readLine()) != null) {
22                System.out.println(x);
23            }
24        } catch (IOException e) {
25            System.out.println("Handle Exception");
26            e.printStackTrace();
27        }
28    }
29 }
```

### Output:

```
Output - readwriteop (run)

run:
Create File: Mostakim.txt
Location: D:\Netbeans\readwriteop\Mostakim.txt
File Size: 0
Check The File
CITY UNIVERSITY
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
Mostakim.txt - Notepad
File Edit Format View Help
CITY UNIVERSITY
```

## 9. Java JDBC Database Connection and Delete Operation

### Code:

```
Connectdb.java X
Source History
1 package connectdb;
2 import java.sql.*;
3 public class Connectdb {
4     public static void delete(int id) {
5         try {
6             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/connectdb", "root", "");
7             String s = "DELETE FROM std_info WHERE id=?";
8             PreparedStatement ps = conn.prepareStatement(s);
9             ps.setInt(1, id);
10            int r = ps.executeUpdate();
11            if (r > 0) {
12                System.out.println("DELETED");
13            } else {
14                System.out.println("ID NOT FOUND");
15            }
16            conn.close();
17        } catch (Exception e) {
18            e.printStackTrace();
19        }
20    }
21    public static void main(String[] args) {
22        try {
23            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/connectdb", "root", "");
24            System.out.println("CONNECTED SUCCESSFULLY");
25            String sq = "INSERT INTO std_info (name, batch) VALUES(?, ?)";
26            PreparedStatement ps = conn.prepareStatement(sq);
27            ps.setString(1, "Mostakim");
28            ps.setString(2, "58");
29            //ps.executeUpdate();
30            System.out.println("INSERTED");
31            conn.close();
32        } catch (SQLException e) {
33            e.printStackTrace();
34        }
35        //delete(1);
36    }
37 }
```

### Output:

Output - connectdb (run)

```
run:
CONNECTED SUCCESSFULLY
INSERTED
BUILD SUCCESSFUL (total time: 1 second)
```




	id	name	batch
<input type="checkbox"/> Edit Copy Delete	2	alhaz	58
<input type="checkbox"/> Edit Copy Delete	3	Mostakim	58



**Delete code:**

```
Connectdb.java x
Source History
1 package connectdb;
2 import java.sql.*;
3 public class Connectdb {
4     public static void delete(int id) {
5         try {
6             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/connectdb", "root", "");
7             String s = "DELETE FROM std_info WHERE id=?";
8             PreparedStatement ps = conn.prepareStatement(s);
9             ps.setInt(1, id);
10            int r = ps.executeUpdate();
11            if (r > 0) {
12                System.out.println("DELETED");
13            } else {
14                System.out.println("ID NOT FOUND");
15            }
16            conn.close();
17        } catch (Exception e) {
18            e.printStackTrace();
19        }
20    }
21    public static void main(String[] args) {
22        try {
23            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/connectdb", "root", "");
24            System.out.println("CONNECTED SUCCESSFULLY");
25            String sq = "INSERT INTO std_info (name, batch) VALUES(?, ?)";
26            PreparedStatement ps = conn.prepareStatement(sq);
27            ps.setString(1, "Mostakim");
28            ps.setString(2, "58");
29            //ps.executeUpdate();
30            System.out.println("INSERTED");
31            conn.close();
32        } catch (SQLException e) {
33            e.printStackTrace();
34        }
35        delete(1);
36    }
37 }
```

**Output:**

				id	name	batch
<input type="checkbox"/>	 Edit	 _Copy	 Delete	3	Mostakim	58

## 10. Program for a Java Server

### Code:

```
server.java X client.java X
Source History
1 package server;
2 import java.io.*;
3 import java.net.*;
4 public class server {
5     public static void main(String[] args) throws IOException {
6         ServerSocket ss = new ServerSocket(5000);
7         System.out.println("Server is Waiting.....");
8         Socket cs = ss.accept();
9         System.out.println("Server is Connected to Client");
10        BufferedReader in = new BufferedReader(new InputStreamReader(cs.getInputStream()));
11        String name = in.readLine();
12        System.out.println("Receive Name: " + name);
13        cs.close();
14        ss.close();
15    }
16 }
```

```
server.java X client.java X
Source History
1 package server;
2 import java.io.*;
3 import java.net.*;
4 public class client {
5     public static void main(String[] args) throws IOException {
6         Socket cs = new Socket("localhost", 5000);
7         BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
8         PrintWriter out = new PrintWriter(cs.getOutputStream(), true);
9         System.out.println("Enter Your Name: ");
10        String name = in.readLine();
11        out.println(name);
12        cs.close();
13    }
14 }
```

### Output:

```
Output
serverclient (run) X serverclient (run) #2 X
run:
Server is Waiting.....
Server is Connected to Client
Receive Name: mostakim
BUILD SUCCESSFUL (total time: 9 seconds)
```

```
Output
serverclient (run) X serverclient (run) #2 X
run:
Enter Your Name:
mostakim
BUILD SUCCESSFUL (total time: 5 seconds)
```