1. Implementing Shape Interface in Java.

Code:

Simplejavaserver.java × client.java × ShapeInterface.java ×

Source History

```java
package com.mycompany.shapeinterface;
interface shape {
    void s_shape();
}
class circle implements shape {
    public void s_shape() {
        System.out.println("The Shape is Circle : ");
    }
}
class triangle implements shape {
    public void s_shape() {
        System.out.println("The Shape is Triangle: ");
    }
}
public class ShapeInterface {
    public static void main(String[] args) {
        circle c1 = new circle();
        c1.s_shape();
        triangle t1 = new triangle();
        t1.s_shape();
    }
}
```

Output:

Output ×

Run (simplejavaserver) × Run (ShapeInterface) ×

```
--- 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

```java
package com.mycompany.javacollection;
import java.util.*;
public class Javacollection {
    public static void main(String[] args) {
        //list
        List<String> book = new ArrayList<>();
        book.add("Java Basic Knowledge");
        book.add("Advanced Java");
        System.out.println("Book Name: " + book);
        System.out.println(book.size());
        book.remove("Advanced Java");
        System.out.println("Books Name After Removing: " + book);
        book.clear();
        System.out.println("After Clearing Name: " + book);

        //set
        Set<String> author = new HashSet<>();
        author.add("Muhammad");
        author.add("Mostakim");
        System.out.println("Author Name: " + author);
        author.remove("Muhammad");
        System.out.println("After Removing Author Name: " + author);

        //queue
        Queue<String> request = new LinkedList<>();
        request.add("Searching Books 1");
        request.add("Searching Books 2");
        request.add("Searching Books 3");
        System.out.println("After Removing Searching: " + request);
        request.clear();
        System.out.println("After Clear: " + request);
        System.out.println(request.size());

        //map
        Map<Integer, String> bookId = new HashMap<>();
        bookId.put(101, "Java");
        bookId.put(102, "Java");
        bookId.put(103, "Java");
        System.out.println(bookId);
    }
```

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:

```java
package javathread;
import java.io.*;
class athread extends Thread {
    String msg;
    public athread(String a) {
        msg = a;
    }
    public void run() {
        System.out.println(msg);
    }
}
public class Javathread {
    public static void main(String[] args) {
        athread a1 = new athread("Thread 1");
        a1.start();
        athread a2 = new athread("Thread 2");
        a2.start();
    }
}
```

Output:

```
run:
Thread 2
Thread 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Polymorphism with Method Overloading

Code:

```java
package polymorphism;
class math {
    int add(int m, int n) {
        return m + n;
    }
    int add(int m, int n, int p) {
        return m + n+ p;
    }
}
public class Polymorphism {
    public static void main(String[] args) {
        math ans = new math();
        System.out.println("Result m+n=" + ans.add(1, 1));
        System.out.println("Result m+n+p=" + ans.add(1, 1, 1));
    }
}
```

Output:

```
Output
Delete Project ×   polymorphism (run) ×
run:
Result m+n=2
Result m+n+p=3
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Inheritance in Java

Code:

```
...va  Javathread.java ×  Server.java ×  client.java ×  Polymorphism.java ×  Inheritance.java ×
Source  History
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:

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

6. Multithreading with Loop in Java

Code:

```java
package threadloop;
import java.io.*;
class athread extends Thread {
    String msg;
    public athread(String a) {
        msg = a;
    }
    public void run() {
        for (int i = 1; i <= 2; i++) {
            System.out.println(msg);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
public class Threadloop {
    public static void main(String[] args) {
        athread a1 = new athread("Thread 1");
        a1.start();
        athread a2 = new athread("Thread 2");
        a2.start();
    }
}
```

Output:

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

7. Serialization in Java

Code:

```java
package serialization;
import java.io.*;
class person implements Serializable {
    public static final long serialVersionUID = 1L;
    String name;
    int id;
    transient String password;
    public person(String s, int i, String p) {
        name = s;
        id = i;
        password = p;
        System.out.println("Default Constructor");
    }
    public void display() {
        System.out.println(name + "\n" + id + "\n" + password);
    }
}
public class Serialization {
    public static void main(String[] args) {
        person p1 = new person("Mostakim", 1051, "PASSWORD NAI");
        p1.display();
        try {
            FileOutputStream fo = new FileOutputStream("person.ser");
            ObjectOutputStream os = new ObjectOutputStream(fo);
            os.writeObject(p1);
            os.close();
            fo.close();
            System.out.println("Object is Serialized");
            FileInputStream fi = new FileInputStream("Person.ser");
            ObjectInputStream os2 = new ObjectInputStream(fi);
            person dp = (person) os2.readObject();
            os2.close();
            fi.close();
            System.out.println("Object is Deserialized");
            dp.display();
        } catch (Exception e) {
            System.out.println("An Error Occurred");
            e.printStackTrace();
        }
    }
```

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:

```java
package readwriteop;
import java.io.*;
public class Readwriteop {
    public static void main(String[] args) {
        try {
            File f1 = new File("Mostakim.txt");
            if (f1.createNewFile()) {
                System.out.println("Create File: " + f1.getName());
                System.out.println("Location: " + f1.getAbsolutePath());
            } else {
                System.out.println("File Already Created");
                System.out.println("Location: " + f1.getAbsolutePath());
            }
            System.out.println("File Size: " + f1.length());
            FileWriter w = new FileWriter(f1, true);
            w.write("CITY UNIVERSITY");
            System.out.println("Check The File");
            w.close();
            BufferedReader b1 = new BufferedReader(new FileReader(f1));
            String x;
            while ((x = b1.readLine()) != null) {
                System.out.println(x);
            }
        } catch (IOException e) {
            System.out.println("Handle Exception");
            e.printStackTrace();
        }
    }
}
```

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

```java
package connectdb;
import java.sql.*;
public class Connectdb {
    public static void delete(int id) {
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/connectdb", "root", "");
            String s = "DELETE FROM std_info WHERE id=?";
            PreparedStatement ps = conn.prepareStatement(s);
            ps.setInt(1, id);
            int r = ps.executeUpdate();
            if (r > 0) {
                System.out.println("DELETED");
            } else {
                System.out.println("ID NOT FOUND");
            }
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/connectdb", "root", "");
            System.out.println("CONNECTED SUCCESSFULLY");
            String sq = "INSERT INTO std_info (name, batch) VALUES(?, ?)";
            PreparedStatement ps = conn.prepareStatement(sq);
            ps.setString(1, "Mostakim");
            ps.setString(2, "58");
            //ps.executeUpdate();
            System.out.println("INSERTED");
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        //delete(1);
    }
}
```

Output:

Output - connectdb (run)

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

				id	name	batch
Edit	Copy	Delete		2	alhaz	58
Edit	Copy	Delete		3	Mostakim	58

Delete code:

Connectdb.java

Source History

```java
package connectdb;
import java.sql.*;
public class Connectdb {
    public static void delete(int id) {
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/connectdb", "root", "");
            String s = "DELETE FROM std_info WHERE id=?";
            PreparedStatement ps = conn.prepareStatement(s);
            ps.setInt(1, id);
            int r = ps.executeUpdate();
            if (r > 0) {
                System.out.println("DELETED");
            } else {
                System.out.println("ID NOT FOUND");
            }
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/connectdb", "root", "");
            System.out.println("CONNECTED SUCCESSFULLY");
            String sq = "INSERT INTO std_info (name, batch) VALUES(?, ?)";
            PreparedStatement ps = conn.prepareStatement(sq);
            ps.setString(1, "Mostakim");
            ps.setString(2, "58");
            //ps.executeUpdate();
            System.out.println("INSERTED");
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        delete(1);
    }
}
```

Output:

		id	name	batch
☐	Edit Copy Delete	3	Mostakim	58

10.Program for a Java Server

Code:

server.java × client.java ×

Source History

```java
package server;
import java.io.*;
import java.net.*;
public class server {
    public static void main(String[] args) throws IOException {
        ServerSocket ss = new ServerSocket(5000);
        System.out.println("Server is Waiting......");
        Socket cs = ss.accept();
        System.out.println("Server is Connected to Client");
        BufferedReader in = new BufferedReader(new InputStreamReader(cs.getInputStream()));
        String name = in.readLine();
        System.out.println("Receive Name: " + name);
        cs.close();
        ss.close();
    }
}
```

server.java × client.java ×

Source History

```java
package server;
import java.io.*;
import java.net.*;
public class client {
    public static void main(String[] args) throws IOException {
        Socket cs = new Socket("localhost", 5000);
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        PrintWriter out = new PrintWriter(cs.getOutputStream(), true);
        System.out.println("Enter Your Name: ");
        String name = in.readLine();
        out.println(name);
        cs.close();
    }
}
```

Output:

Output

serverclient (run) × serverclient (run) #2 ×

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

Output

serverclient (run) × serverclient (run) #2 ×

```
run:
Enter Your Name:
mostakim
BUILD SUCCESSFUL (total time: 5 seconds)
```