Department of Computer Science and Engineering

Lab Manual

CSC592
OOP Using Java

List of Program

1. **Class creation with main method and steps of compilation and Execution.**
2. **Design of Stack andQueue using different types of Linked List for Different Operations.**
3. **Implement Method Overloading and Method Overriding.**
4. **Implement different types of Inheritance with the super () keyword.**
5. **Abstract class and Interface Creation to implement an abstraction**
6. **Package Creation and program using Interface.**
7. **Implement Checked and Unchecked Exceptions through Exception Handling.**
8. **Implement multi-thread application using Thread class and Runnable interface.**
9. **Java applet implementation to draw a Circle inside a Rectangle.**

1. **Aim:** To create a Java Class and Show its Steps of Compilation and Execution.

Step 1: Open a text editor, like Notepad on windows and TextEdit on Mac. Copy the above program and paste it in the text editor.

Step 2: Save the file as **FirstJavaProgram.java**. we should always name the file same as the public class name. Our program's public class name is FirstJavaProgram, so our file name should be **FirstJavaProgram.java**.

Step 3: open **command prompt (cmd) on Windows**, if you are **Mac OS then open Terminal**. To compile the program, type the following command and hit enter.

javac FirstJavaProgram.java

Step 4: After compilation the .java file gets translated into the .class file(byte code). Now we can run the program. To run the program, type the following command and hit enter: **java FirstJavaProgram**

Example

```
public class FirstJavaProgram
{
public static void main(String[] args)
{
System.out.println("This is my first program in java");
}
}
```

public: This makes the main method public that means that we can call the method from outside
the class.
static: We do not need to create object for static methods to run. They can run itself.

void: It does not return anything.
main: It is the method name. This is the entry point method from which the JVM can run your
program.
(String[] args): Used for command line arguments that are passed as strings. We will cover that
in a separate post.
System.out.ptintln: This method prints the contents inside the double quotes into the console
and inserts a newline after.

Output : "This is my First Java Program."

Result : This way a Java Program is created and Compiled and Executed.

2. **Aim :**

To design Stack , Queue and different types of Linked Lists using Java and Implement it for different types of operations.

A. **Source Code :**

```
import java.io.*; class StackArray
{
static int max=10,i,top,ch,item; static int a[]=new int[10];
StackArray()
{
top=-1;
}
public static void main(String args[])throws IOException
{
while((boolean)true)
{
System.out.println("enter 1.Push 2.Pop 3.Display 4.Exit");
try
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
ch=Integer.parseInt(br.readLine());
}
catch(Exception e) { } if(ch==4)
break; else
{
switch(ch)
{
case 1: push();
break;
case 2: pop();
break;
case 3: display();
break;
}}}}
static void push()
{
if(top==max) System.out.println("stack is full"); else
try
{BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("enter the element:"); item=Integer.parseInt(br.readLine());
a[++top]=item;
}
catch(Exception e) { }
}
```

```
static void pop()
{
if(top==-1)
System.out.println("stack is empty"); else
top--;
System.out.println("poped    item:"+a[top]);
}
static void display()
{
System.out.println("elements in stack are:"); for(i=top; i>0; i--)
System.out.println(a[i]);
}
}
```

Output :

```
enter 1.Push 2.Pop 3.Display 4.Exit 1
enter the element:
10
enter 1.Push 2.Pop 3.Display 4.Exit 1
enter the element:
20
enter 1.Push 2.Pop 3.Display 4.Exit 3

elements in stack are:
20
10
enter 1.Push 2.Pop 3.Display 4.Exit 2
poped item:10
enter 1.Push 2.Pop 3.Display 4.Exit 3
elements in stack are:
10
enter 1.Push 2.Pop 3.Display 4.Exit 4
```

2b) Queue operation Using Java

```
import java.io.*;
class QueueArr
{
static int i,front,rear,item,max=5,ch;
static int a[]=new int[5];
QueueArr()
{
front=-1;
rear=-1;
}
public static void main(String args[])throws IOException
{
while((boolean)true)
{
try
{
System.out.println("Select Option 1.insert 2.delete 3.display 4.Exit");
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
ch=Integer.parseInt(br.readLine());
}
catch(Exception e)
{ }
if(ch==4)
break;
else
{
switch(ch)
{
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
display();
break;
}
}
}
}
static void insert()
{
if(rear>=max)
{
System.out.println("Queue is Full");
}
```

```
else{
try
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the Element: ");
item=Integer.parseInt(br.readLine());
}
catch(Exception e)
{}
rear=rear+1;
a[rear]=item;
}
}
static void delete()
{
if(front==-1)
{
System.out.println("Queue is Empty");
}
else
{
front=front+1;
item=a[front];
System.out.println("Deleted Item: "+item);
}
}
static void display()
{
System.out.println("Elements in the Queue are:");
for(int i=front+1; i<=rear; i++)
{
System.out.println(a[i]);
}}}
```

Output:

```
Select Option 1.insert 2.delete 3.display 4.Exit
1
Enter the Element:
12
Select Option 1.insert 2.delete 3.display 4.Exit
1
Enter the Element:
24
Select Option 1.insert 2.delete 3.display 4.Exit
1
Enter the Element:
36
Select Option 1.insert 2.delete 3.display 4.Exit
3
Elements in the Queue are:
12
24
36
Select Option 1.insert 2.delete 3.display 4.Exit
2
Deleted Item: 12
Select Option 1.insert 2.delete 3.display 4.Exit
3
Elements in the Queue are:
24
36
Select Option 1.insert 2.delete 3.display 4.Exit
4
```

2c) Linked List Using Java

```
public class LinkedList {
class Node{
int data;
Node next;
public Node(int data) {
this.data = data;
this.next = null;
}
}
public Node head = null;
publ ic Node tail = null;
public void addNode(int data) {
Node newNode = new Node(data);
if(head == null) {
head = newNode;
tail = newNode;
}
else {
tail.next = newNode;
tail = newNode;
}
}
public void display() {
Node current = head;
if(head == null) {
System.out.println("List is empty");
return;
}
System.out.println("Nodes of singly linked list: ");
while(current != null) {
System.out.print(current.data + " ");
current = current.next;
}
System.out.println();
}
public static void main(String[] args) {
SinglyLinkedList sList = new SinglyLinkedList();
sList.addNode(1);
sList.addNode(2);
sList.addNode(3);
sList.addNode(4);
sList.display();
} }
```

Output:
Nodes of singly linked list:
1 2 3 4

Result : This way Different Data Staructure is created and implimented for operation .

3.

Aim :

To Implement method Overloading and Overriding using Java.

A. Method Overloading :

```
class Calculator
{
int addition(int op1, int op2)
{
return op1+op2;
}
int addition(int op1, int op2, int op3)
{
return op1+op2+op3;
}
}
public class CompileTimePolymorphism
{
public static void main(String args[])
{
Calculator obj = new Calculator();
System.out.println("Addition of two operands is "+obj.addition(10, 20));
System.out.println("Addition of three operands is "+obj.addition(10, 20, 30));
}
```

Output :

Addition of two operands is:30 Addition of two operands is:60

B. Method Overriding :

```
class Human
{
public void eat()
{
System.out.println("Human is eating");
}
}
class Boy extends Human{
public void eat()
{
System.out.println("Boy is eating");
}
public static void main( String args[])
{
Boy obj = new Boy();
obj.eat();
}
}
```

Output :

The Boy is Eatting.

Result :

Hereby both Polymorphism is creaed and methods are implemented .

4.

Aim :

To Implement Different types of inheritence and use of super() keyword.

Single Inheritance

```
class ParentClass{
ParentClass(){
System.out.println("Constructor of Parent");
}
}
class JavaExample extends ParentClass{
JavaExample(){
System.out.println("Constructor of Child");
}
public static void main(String args[]){
new JavaExample();
}
}
```

Output:

```
Constructor of Parent
Constructor of Child
```

With Super

```
class ParentClass
{
ParentClass()
{
System.out.println("Constructor of Parent");
}
void disp(){
System.out.println("Parent Method");
}
}
class JavaExample extends ParentClass
{
JavaExample()
{
System.out.println("Constructor of Child");
}
void disp()
{
System.out.println("Child Method");
super.disp();
}
public static void main(String args[])
{
JavaExample obj = new JavaExample();
obj.disp();
}
}
```

Output:
Constructor of Parent
Constructor of Child
Child Method
Parent Method

Result:

This way different types of Inheritance are achieved and implemented.

5.

Aim :
To Create Abstract Class and Interface and Implement this.

Source Code :

```
interface A
{
void a();

void b();

void c();

void d();

}

abstract class B implements A

{

public void c()

{

System.out.println("I am C");}

}

class M extends B

{
public void a()
{
System.out.println("I am a");
}
public void b()
{
System.out.println("I am b");
}
public void d()
{System.out.println("I am d");
}

}
```

```
class Test

{

public static void main(String args[])

{

A a=new M();

a.a();

a.b();

a.c();

a.d();
}
}
```

Output :

```
I am a.
I am b.
I am c.
I am d.
```

Result:

This way abstract class and interface are created and they are extended and implemented.

6.

Aim :

To create a Package and program using an access specifier .

Source Code :

```
package pack;
public class A
{
public void

msg()

{

System.out.println("Hello");}

}
package mypack;
import pack.A;
class B{
public static void main(String args[])
{
A obj = new A();

obj.msg();

}

}
```

Output :

Hello

Result :

Step to compile:

To Compile: javac -d. B.java

To Run: java pack.B

The -d is a switch that tells the compiler where to put the class file i.e. the it represents destination. The represents the current folder.

This way Java Package is created and imported to another class work with it .

7.

Aim :

To catch checked and un-checked exceptions using exception handling.

Checked Exception:

```
import java.io.*;
class file
{
public static void main(String[] args)
{
try
{
FileReader fi = new FileReader("somefile.txt");
}
catch (FileNotFoundException e)
{
e.printStackTrace();
}
}
```

Unchecked Exception:

Source Code :

Output :

Exception in thread "main" java.lang.ArithmeticException: / by zero at Main.main(Main.java:5)
Java Result: 1

Result :

Exception handling is implemented using
exception keyword to catch un-expected error !.

8.

Aim :

Implementation of multi-thread using Thread class and Runnable interface .

Source Code :

A. Thread Class :

```
class MultithreadingDemo extends Thread
{
public void run()
{
System.out.println("My thread is in running state.");

}
public static void main(String args[])
{
MultithreadingDemo obj=new MultithreadingDemo();
obj.start();

}

}
```

B. Runnable Interface:

```
class MultithreadingDemo implements

Runnable{

public void run(){
System.out.println("My thread is in running state.");

}
public static void main(String args[]){ MultithreadingDemo obj=new
MultithreadingDemo();
Thread tobj =new Thread(obj); tobj.start();

}
}
```

Output :

My Thread is in running state.

Result:

Multi-Threading is used to perform a certain task by dividing it into multiple segments and performing those parallel.

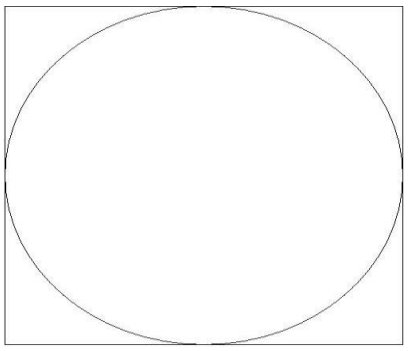
9.

Aim :
To implement java Applet to design a GUI to draw a circle inside a rectangle.

Source Code :

```
import java.applet.Applet; import java.awt.Graphics;

public class rectCircle extends Applet

{

public void paint(Graphics obj)
    {
                        obj.drawRect(100, 100, 500, 500);

                        obj.drawOval(100, 100, 500, 500);

    }

}
```

Output:



Result:

Java applet is a GUI toolkit / framework that was used to design GUI in webpages. This is a old toolkit that is no more used since it was replaced by JavaScript .