GOVERNMENT COLLEGE OF ENGINEERING AND LEATHER TECHNOLOGY

PCA2

Name:-Arpan Bera

Nabakumar Mahata

Joydev Mondal

Stream:-CSE

Subject:-Object Oriented Programming

Year:-3rd

Sem:-5th

A. Multithreading

1. Write a program in Java to print the current thread (The main Thread).

Code in java

```
import java.io.*;
import java.util.*;
public class Test extends Thread {
	public static void main(String[] args)
	{
		Thread t = Thread.currentThread();
		System.out.println("Current thread: "+ t.getName());
		t.setName("Sanju");
		System.out.println("After name change: "+ t.getName());
```

```
                System.out.println("Main thread priority: "+ t.getPriority());
                t.setPriority(MAX_PRIORITY);
                System.out.println("Main thread new priority: "+ t.getPriority());
                for (int i = 0; i < 5; i++) {
                        System.out.println("Main thread");
                }
                Thread ct = new Thread() {
                        public void run()
                        {
                                for (int i = 0; i < 5; i++) {
                                        System.out.println("Child thread");
                                }
                        }
                };
                System.out.println("Child thread priority: "+ ct.getPriority());
                ct.setPriority(MIN_PRIORITY);
                System.out.println("Child thread new priority: "  + ct.getPriority());
                ct.start();
        }
}
class ChildThread extends Thread {
        @Override public void run()
        {
                for (int i = 0; i < 5; i++) {
                        System.out.println("Child thread");
                }
        }
}
```

Output

```
Current thread: main
After name change: Sanju
Main thread priority: 5
Main thread new priority: 10
```

Main thread

Main thread

Main thread

Main thread

Main thread

Child thread priority: 10

Child thread new priority: 1

Child thread

Child thread

Child thread

Child thread

Child thread

2. Create multiple threads (more than 3) extending the Thread class.

Code in Java

```
class CustomThread extends Thread {
    private String threadName;
    public CustomThread(String name) {
        threadName = name;
    }
    @Override
    public void run() {
        try {
         System.out.println(threadName + " is starting.");
            Thread.sleep(1000); // Sleep for 1 second
            System.out.println(threadName + " is finished.");
        } catch (InterruptedException e) {
            System.out.println(threadName + " was interrupted.");
        }
    }
}
public class MultipleThreadsExample {
    public static void main(String[] args) {
        CustomThread thread1 = new CustomThread("Thread 1");
        CustomThread thread2 = new CustomThread("Thread 2");
        CustomThread thread3 = new CustomThread("Thread 3");
        CustomThread thread4 = new CustomThread("Thread 4");
        CustomThread thread5 = new CustomThread("Thread 5");
        thread1.start();
        thread2.start();
        thread3.start();
        thread4.start();
        thread5.start();
    }
}
```

Output

Thread 1 is starting.

Thread 2 is starting.

Thread 3 is starting.

Thread 4 is starting.

Thread 5 is starting.

Thread 1 is finished.

Thread 2 is finished.

Thread 3 is finished.

Thread 4 is finished.

Thread 5 is finished.

3. Create multiple threads (more than 3) by implementing the Runnable Interface.

Code in Java

```
class CustomRunnable implements Runnable {
   private String threadName;
   public CustomRunnable(String name) {
      threadName = name;
   }
   @Override
   public void run() {
      try {
        System.out.println(threadName + " is starting.");
         Thread.sleep(1000);
       System.out.println(threadName + " is finished.");
      } catch (InterruptedException e) {
         System.out.println(threadName + " was interrupted.");
      }
   }
}
public class MultipleThreadsWithRunnable {
   public static void main(String[] args) {
     Runnable task1 = new CustomRunnable("Thread 1");
      Runnable task2 = new CustomRunnable("Thread 2");
      Runnable task3 = new CustomRunnable("Thread 3");
      Runnable task4 = new CustomRunnable("Thread 4");
      Runnable task5 = new CustomRunnable("Thread 5");
      Thread thread1 = new Thread(task1);
      Thread thread2 = new Thread(task2);
      Thread thread3 = new Thread(task3);
      Thread thread4 = new Thread(task4);
      Thread thread5 = new Thread(task5);
      thread1.start();
      thread2.start();
```

```
        thread3.start();
        thread4.start();
        thread5.start();
    }
}
```

Output

```
Thread 1 is starting.
Thread 2 is starting.
Thread 3 is starting.
Thread 4 is starting.
Thread 5 is starting.
Thread 1 is finished.
Thread 2 is finished.
Thread 3 is finished.
Thread 4 is finished.
Thread 5 is finished.
```

B. Exception Handing

2. Use the multiple catch concept.

Code in Java

```
public class MultipleCatchExample {
    public static void main(String[] args) {
        try {
  int[] numbers = new int[3];
            numbers[5] = 10;
            String str = null;
            System.out.println(str.length());
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed.");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Array index is out of bounds.");
        } catch (NullPointerException e) {
            System.out.println("Error: Attempted to use a null object.");
        } catch (Exception e) {
   System.out.println("An unexpected error occurred: " + e.getMessage());
        }
        System.out.println("Program execution continues...");
    }
}
```

Output

```
Error: Array index is out of bounds.
Program execution continues...
```

3. Create and handle your own exception.

Code in Java

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}
public class CustomExceptionExample {
    public static void checkAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Age must be 18 or older.");
        } else {
            System.out.println("Valid age: " + age);
        }
    }
    public static void main(String[] args) {
        try {
            checkAge(15);
            checkAge(20);
        } catch (InvalidAgeException e) {
            System.out.println("Caught exception: " + e.getMessage());
        }
    }
}
```

Output

```
Caught exception: Age must be 18 or older.
Valid age: 20
```

C. Generics

Implement a Generics class in java and use the same for 2 different data types.

Code in Java

```
class Box<T> {
    private T value;
    public Box(T value) {
        this.value = value;
    }
    public T getValue() {
        return value;
    }
    public void setValue(T value) {
        this.value = value;
    }
    public void printType() {
        System.out.println("The type of the value is: " + value.getClass().getName());
    }
}
public class GenericsExample {
    public static void main(String[] args) {
        Box<Integer> integerBox = new Box<>(123);
        System.out.println("Value in integerBox: " + integerBox.getValue());
        integerBox.printType();
        Box<String> stringBox = new Box<>("Hello, Generics!");
        System.out.println("Value in stringBox: " + stringBox.getValue());
        stringBox.printType();
    }
}
```

Output

Value in integerBox: 123

The type of the value is: java.lang.Integer

Value in stringBox: Hello, Generics!

The type of the value is: java.lang.String

D. Autoboxing

Implement the auto boxing feature in Java.

Code in Java

```
public class AutoboxingExample {
   public static void main(String[] args)
   int primitiveInt = 10;
      Integer wrapperInt = primitiveInt;
      System.out.println("Autoboxing:");
      System.out.println("Primitive int: " + primitiveInt);
      System.out.println("Wrapper Integer: " + wrapperInt);
      Integer anotherWrapperInt = 20;
      int anotherPrimitiveInt = anotherWrapperInt;
System.out.println("\nUnboxing:");
      System.out.println("Wrapper Integer: " + anotherWrapperInt);
      System.out.println("Primitive int: " + anotherPrimitiveInt);
      java.util.ArrayList<Integer> intList = new java.util.ArrayList<>();
      intList.add(30);
      System.out.println("\nUsing ArrayList with Autoboxing:");
      System.out.println("ArrayList element: " + intList.get(0));
   }
}
```

Output

```
Autoboxing:
Primitive int: 10
Wrapper Integer: 10

Unboxing:
Wrapper Integer: 20
Primitive int: 20

Using ArrayList with Autoboxing:
ArrayList element: 30
```

E. Applet Programming

1. Implement an Simple Applet and run the same with applet viewer.

Code in Java

```
import java.applet.Applet;
import java.awt.Graphics;
public class SimpleApplet extends Applet {
   public void init() {
      setSize(300, 200);
   }
   public void paint(Graphics g) {
      g.drawString("Hello World", 50, 100);
   }
}
```

Output

F. Swing

1. Implement a Gui based frame using Swing with at least 5 components.

Code in Java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class SwingGUIExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Swing GUI Example");
        frame.setLayout(new FlowLayout());
        JLabel label = new JLabel("Enter your name:");
        frame.add(label);
        JTextField textField = new JTextField(20);
        frame.add(textField);
        JButton button = new JButton("Submit");
        frame.add(button);
        JCheckBox checkBox = new JCheckBox("Subscribe to newsletter");
        frame.add(checkBox);
        String[] options = {"Select Gender", "Male", "Female", "Other"};
        JComboBox<String> comboBox = new JComboBox<>(options);
        frame.add(comboBox);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String name = textField.getText();
                boolean isSubscribed = checkBox.isSelected();
                String gender = (String) comboBox.getSelectedItem();
                JOptionPane.showMessageDialog(frame,
                        "Name: " + name + "\n" +
                        "Subscribed: " + isSubscribed + "\n" +
                        "Gender: " + gender);
```

```
        }
    });
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 250);
    frame.setVisible(true);
  }
}
```

Output

```
Name: John Doe
Subscribed: true
Gender: Male
```

3. Implement a keyboard event handling program.

Code in Java

```
import javax.swing.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
public class KeyboardEventExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Keyboard Event Handling Example");
        JTextArea textArea = new JTextArea(10, 30);
        textArea.setEditable(false);
        frame.add(new JScrollPane(textArea));
        textArea.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                int keyCode = e.getKeyCode();
                String keyText = KeyEvent.getKeyText(keyCode);
                textArea.append("Key Pressed: " + keyText + "\n");
            }
            @Override
            public void keyReleased(KeyEvent e) {
                int keyCode = e.getKeyCode();
                String keyText = KeyEvent.getKeyText(keyCode);
                textArea.append("Key Released: " + keyText + "\n");
            }
            @Override
            public void keyTyped(KeyEvent e) {
                char keyChar = e.getKeyChar();
                textArea.append("Key Typed: " + keyChar + "\n"); }
        });
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Output

```mathematica
Key Pressed: Enter
Key Typed:
Key Released: Enter
```

```mathematica
Key Pressed: A
Key Typed: A
Key Released: A
```