FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERIG
Department of Computer Engineering

Experiment 9- Based on Multithreading

1. Course Details:

Academic Year	2023 - 24	Estimated Time	Experiment No.9 – 02 Hours
Course & Semester	S.E. (COMP) – Sem. III	Subject Name	Skill based lab Course-OOP with Java
Module No.	05	Chapter Title	Exception Handling and multithreading
Experiment Type	Software Performance	Subject Code	CSL304

Name of Student	Vivian Vijay Ludrick	Roll No.	9914
Date of Performance:	16-10-2023	Date of Submission:	
CO Mapping	CSL304.4 Implement the concept of inheritance, exception handling and multithreading		

Timeline (2)	Preparedness (2)	Effort (3)	Result (3)	Total (10)

Problem statement:

1) Write a java program to create the child thread, comment on the execution of main and child thread.

CODE :

```
public class ChildThreadExample {

public static void main(String[] args) {


    System.out.println("Main Thread starting....");

    Thread childThread = new Thread(()-> {

        System.out.println("Child Thread starting....");

        try {
```

```
            Thread.sleep(2000);


        } catch(InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Child Thread is done");
        });
    childThread.start();
    try {
        childThread.join();
    } catch(InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("Main Thread is done");
  }


}
```

OUTPUT :

```
Main Thread starting....
Child Thread starting....
Child Thread is done
Main Thread is done
```

2) Using above example demonstrate the following methods.
sleep(), join(), getPrioity(), setPriority(), getName(), setName(),getid(),currentThread(), yield(), suspend(), resume().

CODE :

```
public class ThreadExample {


  public static void main(String[] args) throws InterruptedException {
```

```
// Creating a child thread
Thread childThread = new Thread(() -> {
    try {
        // Child thread sleep for 2 seconds
        System.out.println("Child Thread is sleeping.");
        Thread.sleep(2000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
});

// Setting thread name
childThread.setName("Child Thread");

// Getting thread name and priority
System.out.println("Child Thread Name: " + childThread.getName());
System.out.println("Child Thread Priority: " + childThread.getPriority());

// Setting thread priority
childThread.setPriority(Thread.MAX_PRIORITY);

// Getting thread ID
System.out.println("Child Thread ID: " + childThread.getId());

// Starting the child thread
childThread.start();

// Joining the child thread with main thread
childThread.join();

// Getting the current thread
```

```
        Thread currentThread = Thread.currentThread();

        // Getting current thread name and ID
        System.out.println("Current Thread Name: " + currentThread.getName());
        System.out.println("Current Thread ID: " + currentThread.getId());

        // Yielding the current thread
        Thread.yield();

        // Using wait and notify for suspending and resuming thread
        System.out.println("Suspending current thread...");
        synchronized (currentThread) {
            currentThread.wait();
        }

        System.out.println("Resuming current thread...");
        synchronized (currentThread) {
            currentThread.notify();
        }
    }
}
```

OUTPUT :

```
Child Thread Name: Child Thread
Child Thread Priority: 5
Child Thread ID: 11
Child Thread is sleeping.
Current Thread Name: main
Current Thread ID: 1
Suspending current thread...
```

3) Simulate the simultaneous transactions on ‘withdraw’ and ‘deposit’ on bank account. Demonstrate using multithreading.

CODE :

```
import java.lang.*;

import java.util.*;


class BankAccount

{

   private double balance;


   public BankAccount(double initBalance)

   {

      this.balance = initBalance;

   }

   public synchronized double getBalance()

   {

      return balance;

   }


   public synchronized void deposit(double amount)

   {

      balance += amount;

      System.out.println("Deposited Amount : " + amount);

   }

   public synchronized void withdraw(double amount)

   {

      if (balance >= amount)

      {

         balance -= amount;

         System.out.println("Withdrawn Amount : " + amount);

      }

      else

      {
```

```
        System.out.println(" The amount to Withdraw exceeds the Balance \n The Balance is
"+ balance );
      }
   }
}


public class Bank
{
   public static void main(String[] args)
   {
      System.out.println("Welcome to PASJ Bank Account !");
      System.out.println("Enter the amount :");
      Scanner sc = new Scanner(System.in);
      double value = sc.nextDouble();
      BankAccount ba = new BankAccount(value);


      Thread depositThread = new Thread(() -> {
         for (int i=0; i<5; i++)
         {
            ba.deposit(100.00);
            ba.getBalance();
         }
      });
      Thread withdrawThread = new Thread(() -> {
         for (int i=0; i<5; i++)
         {
            ba.withdraw(150.00);
            ba.getBalance();
         }
      });
```

```
        depositThread.start();
        withdrawThread.start();

        try
        {
            depositThread.join();
            withdrawThread.join();
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
        System.out.println("Final Balance :" +ba.getBalance());
    }
}
```

OUTPUT :

```
Welcome to PASJ Bank Account !
Enter the amount :
1000
Deposited Amount : 100.0
Withdrawn Amount : 150.0
Withdrawn Amount : 150.0
Withdrawn Amount : 150.0
Withdrawn Amount : 150.0
Withdrawn Amount : 150.0
Deposited Amount : 100.0
Deposited Amount : 100.0
Deposited Amount : 100.0
Deposited Amount : 100.0
Final Balance :750.0
```