PART-6

Multithreading

GitHub Repository Link: https://github.com/21ce114/JAVA-Practicals.git

Question 1:	Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.
Answer:	**A. by extending Thread class:**

```
/*ID: 21CE114
Name: Harsh Rana
Git Repository Link:
https://github.com/21ce114/JAVA-Practicals.git
AIM :Write a program to create thread which display “Hello World” message.
A. by extending Thread class
B. by using Runnable interface.
*/

public class Practical6_1a extends Thread {

    public void run() {

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

    public static void main(String[] args) {

        Practical6_1a tl = new Practical6_1a();

        tl.start();

    }
}
```

B. by using Runnable interface:

```
public class Practical6_1b implements Runnable {

    public void run() {

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

    public static void main(String[] args) {

        Practical6_1b tl = new Practical6_1b();
```

```
        t1.run();

    }

}
```

Output:

```
PS C:\Users\HARSH\OneDrive\Desktop\JAVA\Part-6>  &
workspaceStorage\2e638450c9604b507addaa3d19f29bf6\r
Hello World
```

Question 2:

Generate 15 random numbers from 1 to 100 and store it in an int array. Write a program to display the numbers stored at odd indexes by thread1 and display numbers stored at even indexes by thread2.

Answer:

```
/*ID: 21CE114
Name: Harsh Rana
Git Repository Link:
https://github.com/21ce114/JAVA-Practicals.git
AIM :Generate 15 random numbers from 1 to 100 and store it in an
int array. Write a program to display the numbers stored at odd
indexes by thread1 and display numbers stored at even indexes
by thread2.
*/

import java.util.Scanner;


class DistributedSummation extends Thread {
    public static int sum = 0;
    public static int assignedNumbers;
    public int startNumber;
    public int endNumber;

    public void setValue(int a, int b) {
        startNumber = a;
        endNumber = b;
    }

    synchronized public void sum() {
        for (int i = startNumber; i < endNumber; i++) {
            sum += i;
        }
    }

    public void run() {
        System.out.println(Thread.currentThread().getName() + " is running");
    }
```

```
}

public class Practical6_2{
    public static void main(String[] args) throws Exception {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the number upto you wanna find sum:");
        int n = scan.nextInt();
        System.out.println("Enter the no. of threads you want to sum" + n + "
nos. :");
        int numberOfThreads = scan.nextInt();
        scan.close();
        int numberTracker = 1;

        DistributedSummation[] t = new DistributedSummation[numberOfThreads];
        for (int i = 0; i < numberOfThreads; i++) {
            t[i] = new DistributedSummation();
        }

        DistributedSummation.assignedNumbers = n / numberOfThreads;
        int remainingNumbers = n % numberOfThreads;
        for (int i = 0; i < numberOfThreads; i++) {
            t[i].start();
            t[i].setValue(numberTracker, DistributedSummation.assignedNumbers
* (i + 1));
            numberTracker = DistributedSummation.assignedNumbers * (i + 1);
        }
        for (int i = 0; i < numberOfThreads; i++) {
            t[i].sum();
        }

        if (remainingNumbers != 0) {
            t[0].setValue(numberTracker + 1, n + 1);
            t[0].sum();
        }
        if (remainingNumbers != 0)
            System.out.println("The sum of the " + n + " numbers using " +
numberOfThreads + " is "
                    + (DistributedSummation.sum + n - remainingNumbers));
        else
            System.out.println("The sum of the " + n + " numbers using " +
numberOfThreads + " is "
                    + (DistributedSummation.sum + n));
    }
}
```

Output:

```
PS C:\Users\HARSH\OneDrive\Desktop\JAVA\Part-6>  c
.exe' '-cp' 'C:\Users\HARSH\AppData\Roaming\Code\U
ctical6_2'
Enter the number upto you wanna find sum:
15
Enter the no. of threads you want to sum15 nos. :
2
The sum of the 15 numbers using 2 is 120
Thread-1 is running
Thread-0 is running
```

Question 3: Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.

Answer:

```
/*ID: 21CE114
Name: Harsh Rana
Git Repository Link:
https://github.com/21ce114/JAVA-Practicals.git
AIM : Write a program to increment the value of one variable
by one and display it after one second using thread using
sleep() method.
*/

import java.util.Scanner;

class Incre extends Thread {

    int n;

    Incre(int n) {

        this.n = n;
    }

    public void run() {

        n += 1;

        try {

            sleep(1000);

        }
        catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Incremented number: " + n);
    }
```

```
}

public class Practical6_3 {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Integer: ");
        int n = sc.nextInt();
        Thread t1 = new Incre(n);
        t1.start();
    }
}
```

Output:

```
PS C:\Users\HARSH\OneDrive\Desktop\JAVA\Part-6>  &
workspaceStorage\2e638450c9604b507addaa3d19f29bf6\r
Enter Integer:
3
Incremented number: 4
```

Question 4:

Write a program to create three threads ‘FIRST’, ‘SECOND’, ‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’ thread to 5(default) and the ‘THIRD’ thread to 7.

Answer:

```
/*ID: 21CE114
Name: Harsh Rana
Git Repository Link:
https://github.com/21ce114/JAVA-Practicals.git
AIM : Write a program to create three threads ‘FIRST’, ‘SECOND’,
‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’
thread to 5(default) and the ‘THIRD’ thread to 7.
*/

public class Practical6_4 extends Thread {

public void run(){

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

public static void main(String args[]){

// creating one thread

Practical6_4 t1 = new Practical6_4();
Practical6_4 t2 = new Practical6_4();
Practical6_4 t3 = new Practical6_4();
```

```
// set the priority

t1.setPriority (3);

t2.setPriority (5);

t3.setPriority (7);

// print the user defined priority
System.out.println("Priority of thread t1 is: " + t1.getPriority());

System.out.println("Priority of thread t2 is: " + t2.getPriority() +
"(default)");

System.out.println("Priority of thread t3 is: " + t3.getPriority());

t1.start();
}

}
```

Output:

```
PS C:\Users\HARSH\OneDrive\Desktop\JAVA\Part-6>  &
workspaceStorage\2e638450c9604b507addaa3d19f29bf6\re
Priority of thread t1 is: 3
Priority of thread t2 is: 5(default)
Priority of thread t3 is: 7
running...
```

Question 5: Write a program to solve producer-consumer problem using thread Synchronization.

Answer:

```
/*ID: 21CE114
Name: Harsh Rana
Git Repository Link:
https://github.com/21ce114/JAVA-Practicals.git
AIM :Write a program to solve producer-consumer problem using thread
Synchronization..
*/

import java.util.LinkedList;

public class Practical6_5 {
    public static void main(String[] args)
        throws InterruptedException
    {

        final PC pc = new PC();
```

```
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run()
            {
                try {
                    pc.produce();
                }
                catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run()
            {
                try {
                    pc.consume();
                }
                catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        t1.start();
        t2.start();

        t1.join();
        t2.join();
    }

    public static class PC {


        LinkedList<Integer> list = new LinkedList<>();
        int capacity = 2;

        public void produce() throws InterruptedException
        {
            int value = 0;
            while (true) {
                synchronized (this)
                {
```

```
                    while (list.size() == capacity)
                        wait();

                    System.out.println("Producer produced-"
                                       + value);

                    list.add(value++);


                    notify();

                    Thread.sleep(1000);
                }
            }
        }

        public void consume() throws InterruptedException
        {
            while (true) {
                synchronized (this)
                {

                    while (list.size() == 0)
                        wait();

                    int val = list.removeFirst();

                    System.out.println("Consumer consumed-"
                                       + val);

                    notify();

                    Thread.sleep(1000);
                }
            }
        }
    }
}
```

Output:

```
PS C:\Users\HARSH\OneDrive\Desktop\JAVA\Part-6>  &
workspaceStorage\2e638450c9604b507addaa3d19f29bf6\r
Producer produced-0
Producer produced-1
Consumer consumed-0
Consumer consumed-1
Producer produced-2
Producer produced-3
Consumer consumed-2
Consumer consumed-3
```