**Day 6: Lab assignment on Threads and socket programming**

Q1.create a simple program that creates and runs 10 threads.

Each thread calculates and prints the multiplication table of a number between one and 10.

Run the program and see how the different threads work in parallel.

1. try program by extending Thread class

1. try program by implementing Runnable interface

Q2.Write a program that establishes the name and priority for 10 threads (Refer Q1)

and then shows information about their status until they finish. The threads will calculate the

multiplication table of a number.

The Thread class saves some information attributes that can help us to identify a thread,

know its status, or control its priority. These attributes are:

ID: This attribute stores a unique identifier for each Thread .

Name: This attribute store the name of Thread .

Priority: This attribute stores the priority of the Thread objects. Threads can have a.

Status: This attribute stores the status of Thread . In Java, Thread can be in

one of these six states: new , runnable , blocked , waiting , time waiting ,or terminated .

Q3. Interrupting a thread

A Java program with more than one execution thread only finishes when the execution of all of

its threads end (more specifically, when all its non-daemon threads end its execution or when

one of the threads use the System.exit() method).

Sometimes, you will need to finish athread, because you want to terminate a program,

or when a user of the program wants to cancel the tasks that a Thread object is doing.

Java provides the interruption mechanism to indicate to a thread that we want to finish it.

One peculiarity of this mechanism is that Thread has to check if it has been interrupted or

not, and it can decide if it responds to the finalization request or not.

Write a program that creates Thread and, after 5 seconds, will force

its finalization using the interruption mechanism.

public class PrimeGenerator extends Thread {

@Override

public void run() {

long number = 1L;

// This loop never ends... until is interrupted

while (true) {

if (isPrime(number)) {

System.out.printf("Number %d is Prime\n", number);

}

// When is interrupted, write a message and ends

if (isInterrupted()) {

System.out.printf("The Prime Generator has been Interrupted\n");

return;

}

number++;

}

}

private boolean isPrime(long number) {

if (number <= 2) {

return true;

}

for (long i = 2; i < number; i++) {

if ((number % i) == 0) {

return false;

}

}

return true;

}

}

Q4. Processing uncontrolled exceptions in a thread

When a checked exception is thrown inside the run() method of a Thread object, we have

to catch and treat them, because the run() method doesn't accept a throws clause. When

an unchecked exception is thrown inside the run() method of a Thread object, the default

behaviour is to write the stack trace in the console and exit the program.

How to handle exception thrown from run method ? refer attached lab 4 code and write suitable code

public class Task implements Runnable {

@Override

public void run() {

// The next instruction always throws and exception

int numero=Integer.parseInt("java");

}

}

Q5. Synchronizing a method,In this lab we will learn how to use synchronization in

Java, that is, the use of the synchronized keyword to control the concurrent access to a

method. Only one execution thread will access one of the methods of an object declared with

the synchronized keyword. If another thread tries to access any method declared with the

synchronized keyword of the same object, it will be suspended until the first thread finishes

the execution of the method.

public class Account {

private double balance;

//getter setter

public void addAmount(double amount) {

//

}

public void subtractAmount(double amount) {

//

}

}

Now same account object is used by Bank and Company class, bank trying to subtract Rs 1000 , 100 time and

Company trying to deposit Rs 1000, 100 time(refer lab code)

Now we need to ensure that total amount must be consistent (Apply synchronization method and block)

Q6. Executing tasks in an executor that returns a result

Callable : This interface has the call() method. In this method, you have to implement the logic of a task. The Callable interface is a parameterized interface, meaning you have to indicate the type of data the call() method will return.

Future : This interface has some methods to obtain the result generated by a Callable object and to manage its state

Consider FactorialCalculator class:

public class FactorialCalculator implements Callable<Integer> {

private Integer number;

public FactorialCalculator(Integer number){

this.number=number;

}

@Override

public Integer call() throws Exception {

//refer attached code

}

}

Now create 10 object of FactorialCalculator and submit it to newFixedThreadPool with 2 thread. You need to get array of

Future<Integer> once you get the processing done print the result.

Q7. Filedownloader use threads for better cpu utilization, we need to create an application to simulate processing of filedownloader consider we have some photos on dropbox (Refer lab 7)

https://www.dropbox.com/photo1.jpg

https://www.dropbox.com/photo2.jpg

...

...

https://www.dropbox.com/photo25.jpg

Modifify the code to use threads so that performance of application can be improved

Q8. Shopping cart application need Synchronizing the purchase method properly otherwise we can have race condition (refer lab 8), Correct synchroniztion issue with the application by properly appling Synchronation