Page No.
Date

Aim- Write a program to demostrate multi-thread application

Theory-

An important property of threads is that they can provide a convenient means of allowing system calls without blocking the entire process in which the thread is running. This property makes threads particularly attractive to use in distributed system as it makes much easier to ~~keep~~ express communication in the form of ~~read~~ maintaining multiple logical connection at the same time.

Multithreaded clients.

→ Seperate threads can be activated
→ Each thread sets up a seperate connection to the server
→ Setting up a connection and reading data from the server can be programmed using the standard system calls, assuming that a blocking call does not suspend the entire process.
→ several connection can be opened simultaneously

Multithread servers

→ One thread, the dispatcher, reads incoming request for an operation
→ Request are sent by clients to a well known end point for the server

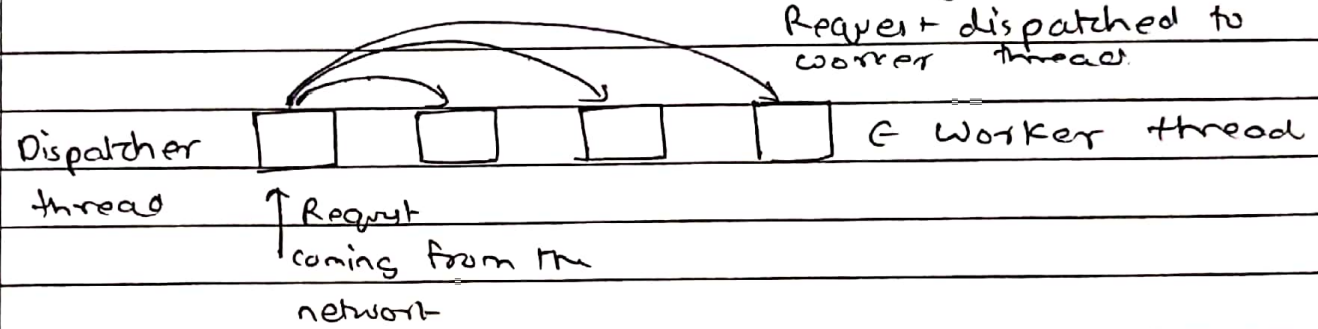
Teacher's Sign.: _____

→ After examining the request, server chooses an idle worker thread and hands it the request

→ Worker proceeds to the tasks

→ If thread is suspended and another thread is selected to be executed

Eg - Dispatcher may be selected to acquire more work. Alternatively, another worker thread can be selected that is now ready to run.



Conclusion -

Threads make it possible to retain the idea of sequential process that make blocking system calls and still achieve parallelism

Blocking system calls make programing easier and parallelism improves performance

Teacher's Sign.: _____

EXPERIMENT 2: MULTI-THREAD APPLICATION(INPUT FROM USER)

```
import java.io.*;
import java.util.Arrays;

class MyThread extends Thread
{
   int[] arr;
        int n;
        MyThread(int[] arr,int n)
        {
                this.arr=arr;
                this.n=n;
        }
        public void run()
        {
      if(n==1){
        Arrays.sort(arr,0,5);
        for(int i =0;i<5;i++)
          System.out.println("Ascending "+arr[i]);
      }
      else if(n==2){
        Arrays.sort(arr,0,5);
        for(int i =4;i>=0;i--)
          System.out.println("Descending "+arr[i]);
      }
      else{
        for(int i =0;i<5;i++){
          if(arr[i]%2==0)
            System.out.println(arr[i]+" is even");
          else
            System.out.println(arr[i]+" is odd");
        }
      }
        }//end of run
}//end of MyThread
class MultiThreadExtra
{
        public static void main(String args[]) throws Exception
        {
                Table t1=new Table();
      int[] arr =new int[5];
      System.out.println("Enter 5 Numbers: ");
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

```
        for(int i =0;i<5;i++)
            arr[i]=Integer.parseInt(br.readLine());

                    MyThread th1= new MyThread(arr,1);
                    MyThread th2= new MyThread(arr,2);
                    MyThread th3= new MyThread(arr,3);
                    th1.start();
                    th2.start();
                    th3.start();
            }
}
```

OUTPUT:

```
D:\D17B-6,8>javac MultiThreadExtra.java

D:\D17B-6,8>java MultiThreadExtra
Enter 5 Numbers:
4
7
8
3
2
Ascending 2
Ascending 3
Ascending 4
Ascending 7
Ascending 8
Descending 8
Descending 7
4 is even
Descending 4
Descending 3
Descending 2
3 is odd
4 is even
7 is odd
8 is even
```