Date:-

Exp:- (7) a.)

Write a program to illustrate the use of Thread class and Runnable interface (Creation of threads)

Thread class
```
import java.io.*;
class Demo extends Thread {
    public Void run()
    {
     System.out.println(" Welcome to CVR ");
    }
   public static Void main (String[] args)
   {
      Demo d = new Demo ();
      g.start ();
   }
}
```
By Implementing ~~Thread~~ Runnable Interface.
```
import java.io.*;
class Demo implements Runnable {
  public static Void main (String[] args) {
    Demo d = new Demo();
    Thread t = new Thread (d, " Yash ");
     t.stat ();
     System.out.println(" t.getName ());
  }
  @Override public void run() {
     System.out.println (" Inside run method ");
  }
}
```

Output 1:-

```
Welcome to CVR
```

~~Yash~~

Output 2:-

```
Yash
Inside run Method
```

Date:-
Exp No:-
(?) b)

Write a progam to illustrate the assignent of thread Priorities.

```
port java. lang. io.*;
ublic class ThreadPriority extends Thread
{ public void run()
 {
 System.out.println(" Inside run Method");
 }
 public static void main(String[] args)
 {
  Thread.currentThread.setPriority(7);
  System.out.println("Priority of main thread:" +
                Thread.currentThread getPriority());
  ThreadPriority tp = new ThreadPriority();
  System.out.println("Priority of thread tp :" + tp.
    getPriority());

 }
}
```

Output:-

```
Inside run Method
Priority of Main Thread: 7
Priority of thread tp: 7
```

Date:- Write a program to Synchronize threads. Use any problem to illustrate the concept

Exp No:- (8) a.)

```
import java.io.*;
import java.util.*;
class Sender {
  public void send (String msg) {
     System.out.println("Sending\t" + msg);
     try{
        Thread.sleep(1000);
     }
     catch (Exception e) {
        System.out.println("Thread Interrupted.");
     }
     System.out.println("\n"+msg + "Sent");
  }
}
class ThreadedSend extends Thread {
   private String msg;
   Sender sender;
   ThreadedSend (String m, Sender obj)
   {
      msg = m;
      sender = obj;
   }
   public void run() {
      synchronized(sender) {
           sender.send(msg);
      }
   }
}
```

```
ass SyncDemo {
public static void main (String args[]) {
    Sender snd = new Sender ();
    ThreadedSend s1 = new ThreadedSend(" Hi", snd);
    ThreadedSend s2 = new ThreadedSend ("Bye", snd);
    S1. start( );
    S2. start( );
    try {
         S1.join ();
        S2.join () ;
     }
     catch (Exeption e)
     {
       System.out. println (" Interrupted ");
      }
  }
}
```

Output:-

```
Sending Hi
  Hi sent
Sending Bye
  Bye sent
```

ate :-

p No:-
i) b.)

Q) Develop a Java Application to demostrate inter thread communication (Producer and Consu problem).

```
port java.util.LinkedList;
ublic class Threadexample {
  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 ~~class~~ 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)
              }
      }
blic 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:-

```
Producer produced - 0
Producer produced - 1
Consumer consumed - 0
Consumer Consumed - 1
Producer Produced - 2
```