Week 8

8) write a program which creates 2 threads, one thread displaying "BMS College of Engineering" once every 10 seconds and another displaying "CSE" once every 2 seconds.

I/P:

```
class DisplayThread extends Thread {
      private String message;
      private int delay;
      private int repetations;

  public DisplayThread (String message, int delay,
  int repetations) {
        this.message = message;
       this.delay = delay;
        this.repetitions = repititions;
  }
  public void run(){
             for (int i=0; i<repitition; i++){
                    System.out.printlm(message);
                    try {
                       Thread.sleep(delay * 1000);
                    } catch (InterruptedException e){
                          e.printStackTrace();
             }
         }
      }
   }

public class ThreadExample {
     public static void main (String args []){
     for int (i=0; repitition < 5; repitition++){
          DisplayThread bmsThread = new Display
Thread ("BMS college of Engineering", 10, 1);

bms Thread.start();

try {
        bms Thread.join();
  } catch (Interrupted Exception e) {
            e. print Stack Trace();
  }
  Display Thread cse Thread = new DisplayT
 -hread ("CSE", 2, 5);

  cse Thread.start();

  try {
        cseThread.join().
     } catch (Interrupted Exception e) {
            e. print Stack Trace();
     }
   }
  }
}
```

OUTPUT:

```
BMS college of Engineering
CSE
CSE
CSE
CSE
CSE
```

write one more

[illegible] 06.02.24

~~[illegible]~~

Inter process communication

Input /code:

```
class Q {
 int n;
 boolean valueset = false;
 synchronized int get() {
 while (! value Set )
 try {
 System.out.println ("\n Consumer waiting\n");
  wait();
  } catch (Interrupted Exception caught ");
  }
  System.out.println ("Got : "+n);
~~synchronized~~
   valueSet = false ;
   System.out.println ("\n Intimate producer\n");
   notify ();
   return n;
 }

Synchronized void put ( int n){
while (valueset )
try {
 System.out.println ("\n producer waiting \n");
 wait ();
 } catch (Interrupted Exception Caught e) {
System.out.println ("Interrupted Exception
caught ");
}
 this.n = n;
 valueSet = true;
 System.out.println ("\n Intimate Consumer\n");
noteify();
}
}

class producer implements Runnable {

Q q;

producer (Q q) {
this.q = q;
new Thread (this, "producer"). start ();
}
public void run () {
        int i=0;
        while (i<15) {
        q. put (i++);
}
}
}

class consumer implements Runnable {
 Q q;
 consumer (Q q) {
 this. q = q;
 new Thread (this, "consumer:" + r);
 i ++;
 }
 }
 }
 class PCFixed {
        public static void main (String args [] ) {
        Q q = new Q();
        new producer (q);
        new consumer (q);
```

```
System.out.println("press Control-c to stop");
}
}
```

OUTPUT

```
Put : 1
Got : 1
put : 2
Got : 2
Put:3
Got : 3
Put :4
Got : 4
Put 5
Got 5
```

[illegible]

classmate
Date
Page

DEBUG DEADLOCK

Input

```
→ class A {
  synchronized void foo (B b){
  String name = Thread. current Thread(). get
 -Name();
  System.out.println (name + "Entered A.foo");
  try {
  Thread. sleep(1000);
  } catch (Exception e){
   System.out.println ("A Interrupted");
   }
   System.out.println (name + "trying to
   call  B.last ()");
    b.last();
    }
    void last(){
    System.out.println ("Inside A.last");
    }
    }
    class Deadlock implements Runnable
    {
     A a = new A();
     B b = new B();
     Deadlock(){
        Thread.current Thread (). setName("m
       -ain Thread");
         Thread  t = new Thread (this, "Racing Thread
         t.start();
         a.foo( b);
          System.out.println ("Back in main
          thread");
          }
public void run(){
b.bar(a);
System.out.println("Back in other thread");
}
public static void main(String args[]){
new Deadlock();
}
}
```

OUTPUT :-

```
Main Thread entered A.foo
Racing Thread entered B.~~foo~~ bar
Main Thread trying to call B.last()
Inside A.last
Back in main thread
Racing Thread trying to call A.last()
Inside A.last
Back in other thread.
```

[illegible] 13.02.24