6/02/24

Lab-8

Q. Write a program which creates two threads, one thread displaying "BMS College of Engineering" once every ten seconds and another displaying "CSE" once every two seconds.

=>

```
class BMSCE implements Runnable{
            public void run(){
                  while(true){
                     try{
                         S.O.P("BMS College of Engineering");
                                Thread.sleep(10000);
                     }catch(InterruptedException e){
                          e.printStackTrace();
                  }
            }
      }
}
class CSE implements Runnable{
            public void run(){
                  while(true){
                     try{
                        S.OP("CSE");
                        Thread.sleep(2000);
                  }catch(InterruptedException e){
                        e.printStackTrace();
            }
      }
}
}
```

```
public class Main {
    public static void main (String []args) {
        Thread t1 = new Thread (new BMSCE());
        Thread t2 = new Thread (new CSE());
            t1.start();
            t2.start();
    }
}
```

Output -

```
BMSCE
CSE
CSE
CSE
CSE
CSE
BMSCE
CSE
CSE
CSE
CSE
CSE
BMSCE
```

Lab-10

Q@ Demonstrate Inter Process Communication and deadlock

IPC

```
class Q {
 int n;
boolean value Set = false;
Sychronized int get () {
 while (! value Set)
 try {
 S.O.P ("\n Consumer waiting \n");
 wait ();
 }
 catch (Interrupted Exception e) {
 S.O.P ("Interrupted Exception caught");
 }
 S.O.P ("Got !" +n);
value Set = false;
 S.O.P ("\n Intimate Producer \n");
Notify ();
 return n;
}
Synchronized void put (int n) {
while (value Set)
try {
S.O.P ("\n Producer waiting \n");
wait ();
}
 Catch (Interrupted Exception e) {
 S.O.P ("Interrupted exception caught");
 }
 this.n = n;
Value set = true;
S.O.P ("Put:" +n);
S.O.P ("\n Intimate Consumer \n");
Notify ();
}}
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"). Start();
 }
public void run (){
        int i =0;
  while (i<15) {
  int r = q. get();
  S.O.P ("Consumed:" +r);
  i++;
  }}
```

```
class PFixed {
public Static void main (String args [ ]) {
Q q = new Q ( );
new Producer (q);
new Consumer (q);
S.O.P ("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
```

(b) Deadlock

```
class A {
Synchronized void foo (B b) {
String name = thread. Current Thread() . getName ();
S.O.P (name + "entered A. foo");
try {
  Thread. sleep(1000);
} Catch (Exception e) {
S.O.P (" A Interrupted");
S.O.P (Name + "trying to call B. last ()");
b. last();
}
Void last () {
S.O.P ("Inside A. last");
  }
}
Class B {
 Synchronized vain bar (A a) {
 Stringname = Thread. Current thread () . getName();
  S.O.P. (name + "entered B. bar");
  try {
  thread - sleep (1000);
 } Catch (Exception e) {
 S.O.P (" B Interrupted");
}
 S.O.P (name + " trying to call A. last ()") };
    a. last ();
  }
Void last () {
          S.O.P ("Inside A. last"); }}
```

```
class Deadlock implements Runnable
{
  A a = new A();
  B b = new B();
 Deadlock(){
 Thread.currentThread().SetName("MainThread");
 thread t = new Thread(this, "Racing thread");
 t.start();
 a.foo(b);
 S.O.P("Back in main thread");
 }
 public void run(){
  b.bar(a);
  S.O.P("Back in other thread");
 }
 Public static void main(String arg[]){
 new Deadlock();
 }
}
```

Output—
MainThread entered A.foo
RacingThread entered B.bar.
MainThread ~~trying to~~ call B.last()
~~Inside A.last~~
~~Back~~ in MainThread
RacingThread trying to call A.last()
Inside A.last
Back in other thread

[illegible]
6.02.12