6/2/24

Lab 8

→ WAP to create 2 threads, BMSCE → one every 10 sec & "CSE" → once every 2 sec

```
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.O.P ("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();
    }
}
```

__/__/____

O/P

BMSCE
CSE
CSE
CSE
CSE
CSE
BMSCE
CSE
CSE
CSE
CSE
CSE
BMSCE

6/2/24

Lab 10

Demonstrate IPC & deadlock.

IPC

```
class Q {
int n;
Boolean valueSet = false;
synchronized int get() {
while (!valueSet)
try {
S.O.P ("\n Consumer waiting\n");
wait();
} catch (InterruptedException e) {
S.O.P (" InterruptedException caught");
}
S.O.P ("Got:" + n);
valueSet = false;
s.O.P ("\n Intimate Producer\n");
notify();
return n;
}
Synchronized void put (int n) {
while (valueSet)
try {
s.O.P ("\n Producer waiting\n");
wait();
} catch (InterruptedException e) {
s.O.P (" InterruptedException caught");
}
this.n = n;
valueSet = true;
```

```
S.o.p ("Put:"+n);
S.o.p ("\n Intimate Consumer 1");
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 PCFixed {
public static void main (String args[]) {
Q q = new Q();
new Producer (q);
new Consumer (q);
S.O.P ("Press Control-C to stop");
}
}
```

O/P

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

__/__/____

DEADLOCK

```
class A {
synchronized void foo (B b) {
String name = Thread.currentThread().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 void bar (A a){
String name= Thread.currentThread().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, "RacingThread");
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 args[]) {
new Deadlock();
}
}
```

O/P

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
Inside A.last
Back in main thread
RacingThread trying to call A.last()
Inside A.last
Back in other thread.
```

8.2.24