13/2/24

Lab 10

```
Class Q {
  int n;
  boolean valueSet = false;
  Synchronized int get() {

    while (!valueSet)
    {
      try {
      System.out.println("\n Consumer waiting\n");
      wait();
      }
      catch (InterruptedException e) {
      System.out.println ("Interrupted Exception
      caught");
      }
    System.out.println ("Got: " + n);
    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 (InterruptedException e)
        {
            System.out.println ("InterruptedException
                                 caught");
        }
    }
    this.n = n;
    valueSet = true;
    System.out.println ("Put: " + n);
    System.out.println ("\n Intimate Consumer\n");
    notify();
  }
}
class Producer implements Runnable {
    Q q;
    Producer (Q q) {
        this.q = q;
        new Thread (this, "Consumer").start();
    }
    public void run() {
        int i=0;
        while (i<15) {
            int r = q.get();
            i++;
        }
    }
}
```

```
class PCFixed {

    public static void main(String args[]) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        new Cor
        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
```

Deadlock

```
class A {

synchronized void foo(B b) {

    String name = Thread.currentThread().getName();
    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 B{

synchronized void bar (A a){

String name = Thread.currentThread().getName();
System.out.println(name + "entered B.bar");

try{
    Thread.sleep(1000);
}
catch(Exception e){
    System.out.println("B Interrupted");
    System.out.println(name + "trying to
    call A.last()");
    a.last();
}
}
void last(){
  System.out.println("Inside A.last");
}
}

class Deadlock implements Runnable
{
        A a= new A();
        B b= new B();

    Deadlock(){
        Thread.currentThread().setName("
        Main 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.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
```