06/02/24 Week - 10 IPC

```
class Q {
 int n;
 boolean valueSet = false;
 Synchronized int get() {
 while (!valueSet)
 try {
      wait();
 } catch (InterruptedException e) {
    System.out.println("InterruptedException caught");}
    System.out.println("Got: " +n);
    valueSet = false;
    notify();
    return n;
    }

Synchronized void put (int n) {
  while (valueSet)
  try{
      wait();
  } catch (InterruptedException e) {
      System.out.println("Put " +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<10){
    int r=q.get();
    i++;
  }
}
}
class P(Fixed){
  public static void main(String args[]){
    Q q=new Q();
    new producer(q);
    new consumer(q);
    System.out.println("Press control-to stop");}
}
```

Output: Put: 0

~~Got: 0~~

Put: 1

Got: 1

Put: 2

Got: 2
Put: 3
Got: 3
Put: 4

Got: 4
Put: 5
Got: 5
Put: 6

Got: 6
Put: 7
Got: 7
Put: 8

Got: 8
Put: 9
Got: 9
Put: 10
Got: 10

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");
        }
```

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
```