27/11/24

WEEK -> 10

Demonstrate inter process communication and deadlock.

Interprocess communication

```
class Q {
    int n;
    boolean value Set = false;
    synchronized int get() {
        while (!value set) {
            try {
                system.out.println("consumer waiting");
                wait();
            } catch (interrupted exception e) {
                System.out.println ("Interrupted exception
                                     caught in get()");
                Thread.currentThread().interrupt();

            }
        }
        System.out.println ("Got :" + n);
        valueset = false;
        System.out.println ("Intimate producer");
        notify ();
        return n;

    }
    synchronized void put (int n) {
        while (value set) {
            try {
                system.out.println ("Producer waiting");
                wait(); }
                Catch (interrupted exception e) {
                System.out.println(" Interrupted exception");
                Thread.current Thread (). interrupt ();

            }
        }
this . n = n;
value Set = true;
System . out . peinthn ("Put : "+n);
System . out . peinthn ("\n Intimate consumer");
notify ();

}
}
class producer implements Runnable {
        Q q;
        peivate static final int max_items = 15;
      Producer ( Q q) {
            this . q = q;
            new thread ( this , "producer"). straart ();

      }
    public void run () {
          int i = 0;
          while (i<max_item) {
                q. put (i++);

          }
      }
  }
class consumer implements Runnalee {
      Q q;
      peivate static final int max-item = 15;
      consumer (Q q) {
            this . q = q;
            new thread (this, "consumer"). start ();

      }
      public void run () {
            int i = 0;
            while (i<max_item) {
              int r = q. get ();
              system . out . peintln ("Consumed" + 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");
    }
}
```

Deadlock : Code.

```
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();
    }
    synchronized void last() {
        System.out.println ("Inside A.last");
    }
}
```

```
class B {
    synchronized void bar (A a) {
            string name = Thread. current Thread (). get Name();
        System . out. peirtln (name + " entered B.bar");

    try {
        thread . sleep (1000);
    } catch (exception e) {
            System. out. peintln ("B interrupted ");

    }
     System.out.peintln (name + "trying to call
                                    A. last()");

      a. last ();
    }
     synchronized void last () {
            System .out. peinth ("Inside B. last ");
       }

   }
class Deadlock implements runnable {
      A   a  = new A ();
      B  b  = new B ();
      deadlock () {
           Thread . current thread (). set Name ("Mainthread");
           Thread . current thread (this, "Racing thread");
           Thread t  = new Thread (this, "Racing thread");

           t. start ();
           Synchronized (a) {
                    a. foo(b);

            }
            system .oUt. println ("Back in main thread");

     }
publiu void run() {
        synchronized (b) {
            b.bar (a);
        }
        System.out.println ("Back in other thread");
    }
    public static void main (String [] args) {
            new Deadlock ();
        }
}
```

27/11/24

WEEK - 10

Demonstrate inter process communication and deadlock.

Output :

```
press control - c to stop.
put : 0
Intimate consumer
producer waiting
 Got : 0
Intimate producer
put : 1
Intimate consumer
producer waiting
consumed : 0
got : 1
Intimate producer
~~custo~~
consumed : 0
got : 1
```

output for deadlock :

```
MainThread entered A.foo
Racing Thread entered B.bar
MainThread trying to call B.last()
Racing Thread trying to call A.last()
```

Seen
[illegible]
27/11/24