2. Design and Develop a Java Program to implement FCFS CPU Scheduling Algorithm

```
import java.util.*;
public class FCFS {
   public static void main(String args[])
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("enter no of process: ");
      int n = sc.nextInt();
      int pid[] = new int[n];   // process ids
      int ar[] = new int[n];    // arrival times
      int bt[] = new int[n];    // burst or execution times
      int ct[] = new int[n];    // completion times
      int ta[] = new int[n];    // turn around times
      int wt[] = new int[n];    // waiting times
      int temp;
      float avgwt=0,avgta=0;
      for(int i = 0; i < n; i++)
      {
         System.out.println("enter process " + (i+1) + " arrival time: ");
         ar[i] = sc.nextInt();
         System.out.println("enter process " + (i+1) + " brust time: ");
         bt[i] = sc.nextInt();
         pid[i] = i+1;
      }
//sorting according to arrival times
      for(int i = 0 ; i <n; i++)
      {
         for(int  j=0;  j < n-(i+1) ; j++)
         {
            if( ar[j] > ar[j+1] )
            {
               temp = ar[j];
               ar[j] = ar[j+1];
               ar[j+1] = temp;
               temp = bt[j];
               bt[j] = bt[j+1];
               bt[j+1] = temp;
               temp = pid[j];
               pid[j] = pid[j+1];
               pid[j+1] = temp;
            }
         }
```

```
    }
// finding completion times
    for(int  i = 0 ; i < n; i++)
    {
      if( i == 0)
      {
        ct[i] = ar[i] + bt[i];
      }
      else
      {
        if( ar[i] > ct[i-1])
        {
          ct[i] = ar[i] + bt[i];
        }
        else
          ct[i] = ct[i-1] + bt[i];
      }
      ta[i] = ct[i] - ar[i] ;         // turnaround time= completion time- arrival time
      wt[i] = ta[i] - bt[i] ;         // waiting time= turnaround time- burst time
      avgwt += wt[i] ;                // total waiting time
      avgta += ta[i] ;               // total turnaround time
    }
    System.out.println("\npid  arrival  brust  complete turn waiting");
    for(int  i = 0 ; i< n;  i++)
    {
      System.out.println(pid[i] + "  \t " + ar[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t"  + wt[i] ) ;
    }
    sc.close();
    System.out.println("\naverage waiting time: "+ (avgwt/n));     // printing average waiting time.
    System.out.println("average turnaround time:"+(avgta/n));    // printing average turnaround
time.
  }
}
```

3. Design and Develop a Java Program to implement SJF CPU Scheduling Algorithm

```
import java.util.*;

public class SJF {
   public static void main(String args[])
   {
      Scanner sc = new Scanner(System.in);
      System.out.println ("enter no of process:");
      int n = sc.nextInt();
      int pid[] = new int[n];
      int at[] = new int[n]; // at means arrival time
      int bt[] = new int[n]; // bt means burst time
      int ct[] = new int[n]; // ct means complete time
      int ta[] = new int[n]; // ta means turn around time
      int wt[] = new int[n];  //wt means waiting time
      int f[] = new int[n];  // f means it is flag it checks process is completed or not
      int st=0, tot=0;
      float avgwt=0, avgta=0;

      for(int i=0;i<n;i++)
      {
         System.out.println ("enter process " + (i+1) + " arrival time:");
         at[i] = sc.nextInt();
         System.out.println ("enter process " + (i+1) + " brust time:");
         bt[i] = sc.nextInt();
         pid[i] = i+1;
         f[i] = 0;
      }
      boolean a = true;
      while(true)
      {
         int c=n, min=999;
         if (tot == n) // total no of process = completed process loop will be terminated
            break;
         for (int i=0; i<n; i++)
         {
            /*
             * If i'th process arrival time <= system time and its flag=0 and burst<min
             * That process will be executed first
             */
            if ((at[i] <= st) && (f[i] == 0) && (bt[i]<min))
```

```
            {
               min=bt[i];
               c=i;
            }
         }
         /* If c==n means c value can not updated because no process arrival time< system time so we
increase the system time */
         if (c==n)
            st++;
         else
         {
            ct[c]=st+bt[c];
            st+=bt[c];
            ta[c]=ct[c]-at[c];
            wt[c]=ta[c]-bt[c];
            f[c]=1;
            tot++;
         }
      }
      System.out.println("\npid  arrival brust  complete turn waiting");
      for(int i=0;i<n;i++)
      {
         avgwt+= wt[i];
         avgta+= ta[i];
         System.out.println(pid[i]+"\t"+at[i]+"\t"+bt[i]+"\t"+ct[i]+"\t"+ta[i]+"\t"+wt[i]);
      }
      System.out.println ("\naverage tat is "+ (float)(avgta/n));
      System.out.println ("average wt is "+ (float)(avgwt/n));
      sc.close();
   }
}
```

4. Design and Develop a Java Program to implement ROUND ROBIN CPU Scheduling Algorithm

```
import java.util.Scanner;
public class RoundRobinExample {
   public static void main(String args[]) {
      int n,x,q_t,count=0,temp,sq=0,b_t[],w_t[],ta_t[],rem_bt[];
      float aw_t=0,ata_t=0;
      b_t = new int[10];
      w_t = new int[10];
      ta_t = new int[10];
      rem_bt = new int[10];
      Scanner sc=new Scanner(System.in);
      System.out.print("Number of processes(maximum 10) = ");
      n = sc.nextInt();
      System.out.print("Enter process burst time\n");
      for (x=0; x<n; x++) {
         System.out.print("P"+x+" = ");
         b_t[x] = sc.nextInt();
         rem_bt[x] = b_t[x];
      }
      System.out.print("Enter the quantum time: ");
      q_t = sc.nextInt();
      while(true) {
         for (x=0,count=0; x<n; x++) {
            temp = q_t;
            if(rem_bt[x] == 0) {
               count++;
               continue;
            }
            if(rem_bt[x]>q_t)
               rem_bt[x]= rem_bt[x] - q_t;
            else if(rem_bt[x]>=0) {
               temp = rem_bt[x];
               rem_bt[x] = 0;
            }
            sq = sq + temp;
            ta_t[x] = sq;
         }
         if(n == count)
            break;
      }
```

```
System.out.print("**********************************************************************
**************");
      System.out.print("\nProcess\t     Burst Time\t      Turnaround Time\t        Waiting Time\n");

System.out.print("**********************************************************************
**************");
      for(x=0; x<n; x++) {
         w_t[x]=ta_t[x]-b_t[x];
         aw_t=aw_t+w_t[x];
         ata_t=ata_t+ta_t[x];
         System.out.println("\n "+(x+1)+"\t "+b_t[x]+"\t\t "+ta_t[x]+"\t\t "+w_t[x]+"\n");
      }
      aw_t=aw_t/n;
      ata_t=ata_t/n;
      System.out.println("\nThe Average waiting Time = "+aw_t+"\n");
      System.out.println("The Average turnaround time = "+ata_t);
   }
}
```

5. Design and Develop a Java Program to implement FIFO Memory Allocations (FIRSTFIT)

```
class FirstFit
{
    static void firstFit(int blockSize[], int m, int processSize[], int n)
   {
      int allocation[] = new int[n];
  for (int i = 0; i < allocation.length; i++)
         allocation[i] = -1;
      for (int i = 0; i < n; i++)
      {
         for (int j = 0; j < m; j++)
         {
            if (blockSize[j] >= processSize[i])
            {
               allocation[i] = j;
               blockSize[j] -= processSize[i];
               break;
            }
         }
      }
      System.out.println("\nProcess No.\tProcess Size\tBlock no.");
      for (int i = 0; i < n; i++)
      {
         System.out.print(" " + (i+1) + "\t\t" +
               processSize[i] + "\t\t");
         if (allocation[i] != -1)
            System.out.print(allocation[i] + 1);
         else
            System.out.print("Not Allocated");
         System.out.println();
      }
   }
   public static void main(String[] args)
   {
      int blockSize[] = {100, 500, 200, 300, 600};
      int processSize[] = {212, 417, 112, 426};
      int m = blockSize.length;
      int n = processSize.length;
      firstFit(blockSize, m, processSize, n);
   }
}
```

6. Design and Develop a Java Program to implement Best Fit Memory Allocations

```
public class BestFit
{
   static void bestFit(int blockSize[], int m, int processSize[],int n)
   {

 int allocation[] = new int[n];
for (int i = 0; i < allocation.length; i++)
          allocation[i] = -1;
       for (int i=0; i<n; i++)
       {
          int bestIdx = -1;
          for (int j=0; j<m; j++)
          {
             if (blockSize[j] >= processSize[i])
             {
                if (bestIdx == -1)
                   bestIdx = j;
                else if (blockSize[bestIdx] > blockSize[j])
                   bestIdx = j;
             }
          }
          if (bestIdx != -1)
          {
             allocation[i] = bestIdx;
             blockSize[bestIdx] -= processSize[i];
          }
       }
       System.out.println("\nProcess No.\tProcess Size\tBlock no.");
       for (int i = 0; i < n; i++)
       {
          System.out.print("   " + (i+1) + "\t\t" + processSize[i] + "\t\t");
          if (allocation[i] != -1)
             System.out.print(allocation[i] + 1);
          else
             System.out.print("Not Allocated");
          System.out.println();
       }
   }
   public static void main(String[] args)
   {
      int blockSize[] = {100, 500, 200, 300, 600};
```

```
        int processSize[] = {212, 417, 112, 426};
        int m = blockSize.length;
        int n = processSize.length;
        bestFit(blockSize, m, processSize, n);
    }
}
```

7. Design and Develop a Java Program to implement Worst Fit Memory Allocations

```
public class WorstFit
{
	static void worstFit(int blockSize[], int m, int processSize[],int n)
	{
		int allocation[] = new int[n];
		for (int i = 0; i < allocation.length; i++)
			allocation[i] = -1;
		for (int i=0; i<n; i++)
		{
			int wstIdx = -1;
			for (int j=0; j<m; j++)
			{
				if (blockSize[j] >= processSize[i])
				{
					if (wstIdx == -1)
						wstIdx = j;
					else if (blockSize[wstIdx] < blockSize[j])
						wstIdx = j;
				}
			}
			if (wstIdx != -1)
			{
				allocation[i] = wstIdx;
				blockSize[wstIdx] -= processSize[i];
			}
		}
		System.out.println("\nProcess No.\tProcess Size\tBlock no.");
		for (int i = 0; i < n; i++)
		{
			System.out.print(" " + (i+1) + "\t\t" + processSize[i] + "\t\t");
			if (allocation[i] != -1)
				System.out.print(allocation[i] + 1);
			else
				System.out.print("Not Allocated");
			System.out.println();
		}
	}
	public static void main(String[] args)
	{
		int blockSize[] = {100, 500, 200, 300, 600};
		int processSize[] = {212, 417, 112, 426};
```

```
                int m = blockSize.length;
                int n = processSize.length;
                worstFit(blockSize, m, processSize, n);
        }
}
```

8. Design and Develop a Java Program to implement Bankers Algorithm Example

```
import java.util.*;
import java.io.*;
import java.util.Scanner;
class BankersAlgoExample
{
   static void findNeedValue(int needArray[][], int maxArray[][], int allocationArray[][], int
totalProcess, int totalResources)
   {
      for (int i = 0 ; i < totalProcess ; i++){    // for each process
         for (int j = 0 ; j < totalResources ; j++){  //for each resource
            needArray[i][j] = maxArray[i][j] - allocationArray[i][j];
         }
      }
   }
   static boolean checkSafeSystem(int processes[], int availableArray[], int maxArray[][], int
allocationArray[][], int totalProcess, int totalResources)
   {
      int [][]needArray = new int[totalProcess][totalResources];
      findNeedValue(needArray, maxArray, allocationArray, totalProcess, totalResources);
      boolean []finishProcesses = new boolean[totalProcess];
      int []safeSequenceArray = new int[totalProcess];
      int []workArray = new int[totalResources];
      for (int i = 0; i < totalResources ; i++)
         workArray[i] = availableArray[i];
      int counter = 0;
      while (counter < totalProcess)
      {
         boolean foundSafeSystem = false;
         for (int m = 0; m < totalProcess; m++)
         {
            if (finishProcesses[m] == false)       // when process is not finished
            {
               int j;
                for (j = 0; j < totalResources; j++)
                  if (needArray[m][j] > workArray[j])
                     break;
               if (j == totalResources)
               {
                  for (int k = 0 ; k < totalResources ; k++)
                     workArray[k] += allocationArray[m][k];
                  safeSequenceArray[counter++] = m;
```

```
                finishProcesses[m] = true;
                foundSafeSystem = true;
              }
            }
          }
           if (foundSafeSystem == false)
          {
            System.out.print("The system is not in the safe state because lack of resources");
            return false;
          }
        }
        System.out.print("The system is in safe sequence and the sequence is as follows: ");
        for (int i = 0; i < totalProcess ; i++)
          System.out.print("P"+safeSequenceArray[i] + " ");
         return true;
      }
      public static void main(String[] args)
      {
        int numberOfProcesses, numberOfResources;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter total number of processes");
        numberOfProcesses = sc.nextInt();
        System.out.println("Enter total number of resources");
        numberOfResources = sc.nextInt();
        int processes[] = new int[numberOfProcesses];
        for(int i = 0; i < numberOfProcesses; i++){
          processes[i] = i;
        }
        int availableArray[] = new int[numberOfResources];
        for( int i = 0; i < numberOfResources; i++){
          System.out.println("Enter the availability of resource"+ i +": ");
          availableArray[i] = sc.nextInt();
        }
        int maxArray[][] = new int[numberOfProcesses][numberOfResources];
        for( int i = 0; i < numberOfProcesses; i++){
          for( int j = 0; j < numberOfResources; j++){
            System.out.println("Enter the maximum resource"+ j +" that can be allocated to process"+ i
+": ");
            maxArray[i][j] = sc.nextInt();
          }
        }
        int allocationArray[][] = new int[numberOfProcesses][numberOfResources];
```

```
      for( int i = 0; i < numberOfProcesses; i++){
         for( int j = 0; j < numberOfResources; j++){
            System.out.println("How many instances of resource"+ j +" are allocated to process"+ i +"?
");
            allocationArray[i][j] = sc.nextInt();
         }
      }
      checkSafeSystem(processes, availableArray, maxArray, allocationArray, numberOfProcesses,
numberOfResources);
   }
}
```

1. Design and Develop a Java Program to implement Reader Writers Problem

```
import java.util.concurrent.Semaphore;
class ReaderWritersProblem {
static Semaphore readLock = new Semaphore(1);
   static Semaphore writeLock = new Semaphore(1);
   static int readCount = 0;
   static class Read implements Runnable {
      public void run() {
         try {
            readLock.acquire();
            readCount++;
            if (readCount == 1) {
               writeLock.acquire();
            }
            readLock.release();
            System.out.println("Thread "+Thread.currentThread().getName() + " is READING");
            Thread.sleep(1000);
            System.out.println("Thread "+Thread.currentThread().getName() + " has FINISHED
READING");
            readLock.acquire();
            readCount--;
            if(readCount == 0) {
               writeLock.release();
            }
            readLock.release();
         } catch (InterruptedException e) {
            System.out.println(e.getMessage());
         }
      }
   }
   static class Write implements Runnable {
      public void run() {
         try {
            writeLock.acquire();
            System.out.println("Thread "+Thread.currentThread().getName() + " is WRITING");
            Thread.sleep(2500);
            System.out.println("Thread "+Thread.currentThread().getName() + " has finished
WRITING");
            writeLock.release();
         } catch (InterruptedException e) {
            System.out.println(e.getMessage());
         }
```

```
        }
    }
    public static void main(String[] args) throws Exception {
        Read read = new Read();
        Write write = new Write();
        Thread t1 = new Thread(read);
        t1.setName("thread1");
        Thread t2 = new Thread(read);
        t2.setName("thread2");
        Thread t3 = new Thread(write);
        t3.setName("thread3");
        Thread t4 = new Thread(read);
        t4.setName("thread4");
        t1.start();
        t3.start();
        t2.start();
        t4.start();
    }
}
```

9. Design and Develop a Java Program to implement FIFO page replacement algorithm

```
import java.io.*;
class FIFO
{

	public static void main(String args[]) throws IOException
	{

		int n;
		int f;

		float rat;
		BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
		System.out.println("Enter the number of FRAMES :");
		f=Integer.parseInt(br.readLine());
		int fifo[]=new int[f];
		System.out.println("Enter the number of INPUTS :");
		n=Integer.parseInt(br.readLine());
		int inp[]=new int[n];
		System.out.println("Enter INPUT:");
		for(int i=0;i<n;i++)
		inp[i]=Integer.parseInt(br.readLine());
		System.out.println("----------------------");
		for(int i=0;i<f;i++)
			fifo[i]=-1;
		int Hit=0;
		int Fault=0;
		int j=0;
		boolean check;
		for(int i=0;i<n;i++)
		{
			check=false;


				for(int k=0;k<f;k++)
				if(fifo[k]==inp[i])
				{
					check=true;
					Hit=Hit+1;
				}
				if(check==false)
```

```
                {
                    fifo[j]=inp[i];
                    j++;
                    if(j>=f)
                    j=0;
                    Fault=Fault+1;
                }

            }
            rat = (float)Hit/(float)n;
            System.out.println("HIT:"+Hit+"  FAULT:"+Fault+"   HIT RATIO:"+rat);
        }
}
```

10. Design and Develop a Java Program to implement Optimal page replacement algorithm

```
import java.util.*;
class PR3 {
static int check2(int p[], int pf[], int count[], int j, int s) {
      int k;
      for (int i = 0; i < s; ++i) {
         count[i] = 0;
      }
      for (k = 0; k < s; ++k) {
         loop:
         for (int i = j - 1;; --i) {
            ++count[k];
            if (p[i] == pf[k]) {
               break loop;
            }
         }
//System.out.println("count = "+count[k]);
      }
      int max = count[0];
      int pos = 0;
      for (int l = 0; l < s; ++l) {
         if (count[l] > max) {
            max = count[l];
            pos = l;
         }
      }
      return pos;
   }
   static int check(int p[], int pf[], int count[], int j, int s) {
      int k;
      for (int i = 0; i < s; ++i) {
         count[i] = 0;
      }
      int found = 0;
      for (k = 0; k < s; ++k) {
         loop:
         for (int i = j + 1; i < p.length; ++i) {
            ++count[k];
            if (p[i] == pf[k]) {
               ++found;
               break loop;
            }
         }
      }
      int pos;
      if (found < (s - 1)) {
```

```
      System.out.println();
     System.out.println("More than 1 used pages not required for furthur operation.\nSwapping LRU  page.");
      pos = check2(p, pf, count, j, s);
    } else {
      int max = count[0];
      pos = 0;
      for (int l = 0; l < s; ++l) {
        if (count[l] > max) {
          max = count[l];
          pos = l;
        }
      }
    }
    return pos;
  }
  static boolean check1(int p[], int x, int n) {
    for (int i = 0; i < n; ++i) {
      if (p[i] == x) {
        return true;
      }
    }
    return false;
  }
  public static void main(String args[]) {
    Scanner scr = new Scanner(System.in);
    System.out.println("Enter the number of page entries");
    int n = scr.nextInt();
    int fault = 0, hit = 0;
    int p[] = new int[n];
    System.out.println("Enter the pages ");
    for (int i = 0; i < n; ++i) {
      p[i] = scr.nextInt();
    }
    System.out.print("The page table entries are ");
    for (int i = 0; i < n; ++i) {
      System.out.print(" " + p[i]);
    }
    System.out.println();
    int s = 3;
    System.out.println("Do you wish to set a the number of page frames?(1=yes/0=no)");
    int ch = scr.nextInt();

    if (ch == 1) {
      System.out.println("Enter the number of page frames ");
      s = scr.nextInt();
      System.out.println("The number of page frames set to " + s);
```

```
} else {
   System.out.println("The default number of page frames are " + s);
}
int count[] = new int[s];
int pf[] = new int[s];
int t = 0;
int incr = 0;
for (int i = 0; i < s || (t != s); ++i) {
   if (check1(pf, p[i], t)) {
      ++incr;
      System.out.println();
      for (int y = 0; y < t; ++y) {
         System.out.print(pf[y] + " ");
      }
      System.out.print(" Hit");
      ++hit;
   } else {
      ++incr;
      ++t;
      pf[t - 1] = p[i];
      System.out.println();
      for (int y = 0; y < t; ++y) {
         System.out.print(pf[y] + " ");
      }
      System.out.print(" F");
      ++fault;
   }
}
for (int j = incr; j < n; ++j) {
   if (check1(pf, p[j], s)) {
      ++hit;
      System.out.println();
      for (int k = 0; k < s; ++k) {
         System.out.print(pf[k] + " ");
      }
      System.out.print(" H");
   } else {
      int choice = check(p, pf, count, j, s);
      ++fault;
      pf[choice] = p[j];
      System.out.println();
      for (int k = 0; k < s; ++k) {
         System.out.print(pf[k] + " ");
      }
      System.out.print(" F");
   }
```

```
        }
        System.out.println();
        System.out.println("Page faults= " + fault + " Page hits= " + hit);
    }
}
```

11. Design and Develop a Java Program to implement LRU page replacement algorithm

```
import java.io.*;
import java.util.*;
public class LRU {
   public static void main(String[] args) throws IOException
   {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int frames,pointer = 0, hit = 0, fault = 0,ref_len;
      Boolean isFull = false;
      int buffer[];
      ArrayList<Integer> stack = new ArrayList<Integer>();
      int reference[];
      int mem_layout[][];

      System.out.println("Please enter the number of Frames: ");
      frames = Integer.parseInt(br.readLine());

      System.out.println("Please enter the length of the Reference string: ");
      ref_len = Integer.parseInt(br.readLine());

      reference = new int[ref_len];
      mem_layout = new int[ref_len][frames];
      buffer = new int[frames];
      for(int j = 0; j < frames; j++)
            buffer[j] = -1;

      System.out.println("Please enter the reference string: ");
      for(int i = 0; i < ref_len; i++)
      {
        reference[i] = Integer.parseInt(br.readLine());
      }
      System.out.println();
      for(int i = 0; i < ref_len; i++)
      {
        if(stack.contains(reference[i]))
        {
         stack.remove(stack.indexOf(reference[i]));
        }
        stack.add(reference[i]);
        int search = -1;
        for(int j = 0; j < frames; j++)
        {
```

```
        if(buffer[j] == reference[i])
        {
           search = j;
           hit++;
           break;
        }
     }
     if(search == -1)
     {
      if(isFull)
      {
       int min_loc = ref_len;
           for(int j = 0; j < frames; j++)
           {
            if(stack.contains(buffer[j]))
              {
                 int temp = stack.indexOf(buffer[j]);
                 if(temp < min_loc)
                 {
                    min_loc = temp;
                    pointer = j;
                 }
              }
           }
      }
        buffer[pointer] = reference[i];
        fault++;
        pointer++;
        if(pointer == frames)
        {
         pointer = 0;
         isFull = true;
        }
     }
     for(int j = 0; j < frames; j++)
        mem_layout[i][j] = buffer[j];
  }

  for(int i = 0; i < frames; i++)
  {
     for(int j = 0; j < ref_len; j++)
        System.out.printf("%3d ",mem_layout[j][i]);
```

```
            System.out.println();
        }

        System.out.println("The number of Hits: " + hit);
        System.out.println("Hit Ratio: " + (float)((float)hit/ref_len));
        System.out.println("The number of Faults: " + fault);
    }

}
```

12. Design and Develop a Java Program to implement FCFS Disk Scheduling Algorithm

```
class FCFS
{
static int size = 8;
static void FCFS(int arr[], int head)
{
        int seek_count = 0;
        int distance, cur_track;
        for (int i = 0; i < size; i++)
        {
                cur_track = arr[i];
                distance = Math.abs(cur_track - head);
                seek_count += distance;
                head = cur_track;
        }
        System.out.println("Total number of " + "seek operations = " +seek_count);
        System.out.println("Seek Sequence is");

        for (int i = 0; i < size; i++)
        {
                System.out.println(arr[i]);
        }
}
public static void main(String[] args)
{
        // request array
        int arr[] = { 176, 79, 34, 60,
                                92, 11, 41, 114 };
        int head = 50;

        FCFS(arr, head);
}
}
```

```
import java.util.LinkedList;

public class Threadexample {

	public static void main(String[] args)throws
		InterruptedException

	{

		final PC pc = new PC();

		Thread t1 = new Thread(new Runnable() {
			@Override

			public void run()

			{

				try {
					pc.produce();
				}


				catch (InterruptedException e) {
					e.printStackTrace();

				}

			}

		});

		Thread t2 = new Thread(new Runnable() {
			@Override

			public void run()

			{

				try {
					pc.consume();
				}


				catch (InterruptedException e) {
					e.printStackTrace();

				}

			}

		});

		t1.start();

		t2.start();

		t1.join();

		t2.join();

	}
public static class PC {


		LinkedList<Integer> list = new LinkedList<>();

		int capacity= 2;
```

```
public void produce() throws InterruptedException
{
        int value = 0;
        while (true) {
                synchronized (this)
                {
                        while (list.size() == capacity)wait();
                        System.out.println("Producer produced-" + value);list.add(value++);
                        notify();
                        Thread.sleep(1000);
                }}}
                public void consume() throws InterruptedException
{
        while (true) {
                synchronized (this)
                {
                while (list.size() == 0)wait();
                        int val = list.removeFirst(); System.out.println("Consumer
                        consumed-"+ val);notify();
                        Thread.sleep(1000);
                }}}
                }
```

Output:

Producer produced:0

Producer produced :1

Consumer consumed :0

Consumer consumed:1

Producer produced :2

Producer produced:3

Consumer consumed:2

Consumer consumed:3