**Hands On Lab 5**

## **[Inter-thread communication](https://personales.unican.es/corcuerp/java/Labs/LAB_23.htm" \l "Exercise_1)**

* 1. **[Producer-Consumer without inter-thread communication](https://personales.unican.es/corcuerp/java/Labs/LAB_23.htm" \l "5.1)**
  2. **[Producer-Consumer with inter-thread communication](https://personales.unican.es/corcuerp/java/Labs/LAB_23.htm" \l "5.2)**

### **(5.1) Producer-Consumer without inter-thread communication**

1. Create a new Java project

* Select **File->New Java Project (Alt+Shift+N)**. The **New Java Project** dialog box appears.
* For the Project Name field, type in **ProducerConsumerUnsynchronized**as project name.
* Choose ‘Use default JRE (11….). Make sure you have set your default JRE set to version 11 and the appropriate compliance settings as well. Deselect ‘Create module-info.java file’ if already selected. Click Next.Click Finish.
* Observe that **ProducerConsumerUnsynchronized**project appears
* Right click on the project and select **File**->**New Class.**
* Enter in a package name of your choosing or use the default. Type in **ProducerConsumerUnsynchronized**as the class name. The main method stub should also be created.

The IDE generated **ProducerConsumerUnsynchronized .java** is displayed in the source editor window of STS IDE.

2. Modify the IDE generated **ProducerConsumerUnsynchronized.java**as shown in Code-5.11 below.

|  |
| --- |
| public class ProducerConsumerUnsynchronized {      public static void main(String[] args) {                 CubbyHole c = new CubbyHole();                 Producer p1 = new Producer(c, 1);         Consumer c1 = new Consumer(c, 1);          p1.start();         c1.start();     } } |

Code-5.11: ProducerConsumerUnsynchronized.java  
  
3. Write **CubbyHole.java** as shown in Code-5.12 below.  Study the code by paying special attention to the bold fonted parts.

|  |
| --- |
| **// Unsynchronized CubbyHole. // // Results are unpredictable; a number may be read before a number has // been produced or multiple numbers may be produced with only one or // two being read adding synchronization ensures that a number is first // produced, then read in the correct order.**  public class CubbyHole {     private int contents;     private boolean available = false;      public int get() {         available = false;         return contents;     }      public void put(int value) {         contents = value;         available = true;     } } |

Code-5.12: CubbyHole.java  
  
4. Write **Producer.java** as shown in Code-5.13 below.

|  |
| --- |
| public class Producer extends Thread {     private CubbyHole cubbyhole;     private int number;      public Producer(CubbyHole c, int number) {         cubbyhole = c;         this.number = number;     }      public void run() {         for (int i = 0; i < 10; i++) {             cubbyhole.put(i);             System.out.println("Producer #" + this.number                                + " put: " + i);             try {                 sleep((int)(Math.random() \* 100));             } catch (InterruptedException e) { }         }     } } |

Code-5.13: Producer.java  
  
5. Write **Consumer.java** as shown in Code-5.14 below.

|  |
| --- |
| public class Consumer extends Thread {     private CubbyHole cubbyhole;     private int number;      public Consumer(CubbyHole c, int number) {         cubbyhole = c;         this.number = number;     }      public void run() {         int value = 0;         for (int i = 0; i < 10; i++) {             value = cubbyhole.get();             System.out.println("Consumer #" + this.number                                + " got: " + value);         }     } } |

Code-5.14: Consumer.java  
  
6. Build and run the project

* Right click **ProducerConsumerUnsynchronized**project and select **Run**.
* Observe the result in the **Output**window. (Figure-5.15 below)

|  |
| --- |
| Producer #1 put: 0 Consumer #1 got: 0 Consumer #1 got: 0 Consumer #1 got: 0 Consumer #1 got: 0 Consumer #1 got: 0 Consumer #1 got: 0 Consumer #1 got: 0 Consumer #1 got: 0 Consumer #1 got: 0 Consumer #1 got: 0 Producer #1 put: 1 Producer #1 put: 2 Producer #1 put: 3 Producer #1 put: 4 Producer #1 put: 5 Producer #1 put: 6 Producer #1 put: 7 Producer #1 put: 8 Producer #1 put: 9 |

Figure-5.15: Result of running ProducerConsumerUnsynchronized application

### **(5.2) Producer-Consumer with inter-thread communication**

1. Create a new Java project

* Select **File->New Java Project (Alt+Shift+N)**. The **New Java Project** dialog box appears.
* For the Project Name field, type in **ProducerConsumerSynchronized**as project name.
* Choose ‘Use default JRE (11….). Make sure you have set your default JRE set to version 11 and the appropriate compliance settings as well. Deselect ‘Create module-info.java file’ if already selected. Click Next.Click Finish.
* Observe that **ProducerConsumerSynchronized**project appears
* Right click on the project and select **File**->**New Class.**
* Enter in a package name of your choosing or use the default. Type in **ProducerConsumerSynchronized**as the class name. The main method stub should also be created.

The IDE generated **ProducerConsumerSynchronized .java** is displayed in the source editor window of STS IDE.

2. Modify the IDE generated **ProducerConsumerSynchronized.java**as shown in Code-5.21 below.

|  |
| --- |
| public class ProducerConsumerSynchronized {      public static void main(String[] args) {                 CubbyHole c = new CubbyHole();                 Producer p1 = new Producer(c, 1);         Consumer c1 = new Consumer(c, 1);                 p1.start();         c1.start();     } } |

Code-5.21: ProducerConsumerSynchronized.java  
  
3. Write **CubbyHole.java** as shown in Code-5.22 below.  Study the code by paying special attention to the bold fonted parts.

|  |
| --- |
| public class CubbyHole {      private int contents;     **private boolean available = false;**      public **synchronized**int get(int who) {         while (available == false) {             try {                 **wait();**             } catch (InterruptedException e) { }         }         available = false;         System.out.format("Consumer %d got: %d%n", who, contents);         **notifyAll();**         return contents;     }      public synchronized void put(int who, int value) {         while (available == true) {             try {                 **wait();**             } catch (InterruptedException e) { }         }         contents = value;         available = true;         System.out.format("Producer %d put: %d%n", who, contents);         **notifyAll();**     } } |

Code-5.22: CubbyHole.java  
  
4. Write**Producer.java** as shown in Code-5.23 below.

|  |
| --- |
| public class Producer extends Thread {      private CubbyHole cubbyhole;     private int number;      public Producer(CubbyHole c, int number) {         cubbyhole = c;         this.number = number;     }      public void run() {         for (int i = 0; i < 10; i++) {             cubbyhole.put(number, i);             try {                 sleep((int)(Math.random() \* 100));             } catch (InterruptedException e) { }         }     } } |

Code-5.23: Producer.java  
  
5. Write **Consumer.java** as shown in Code-5.24 below.

|  |
| --- |
| public class Consumer extends Thread {     private CubbyHole cubbyhole;     private int number;      public Consumer(CubbyHole c, int number) {         cubbyhole = c;         this.number = number;     }      public void run() {         int value = 0;         for (int i = 0; i < 10; i++) {             value = cubbyhole.get(number);         }     } } |

Code-5.24: Consumer.java  
  
6. Build and run the project

* Right click **ProducerConsumerSynchronized**project and select **Run**.
* Observe the result in the **Output**window. (Figure-5.25 below)

|  |
| --- |
| Producer 1 put: 0 Consumer 1 got: 0 Producer 1 put: 1 Consumer 1 got: 1 Producer 1 put: 2 Consumer 1 got: 2 Producer 1 put: 3 Consumer 1 got: 3 Producer 1 put: 4 Consumer 1 got: 4 Producer 1 put: 5 Consumer 1 got: 5 Producer 1 put: 6 Consumer 1 got: 6 Producer 1 put: 7 Consumer 1 got: 7 Producer 1 put: 8 Consumer 1 got: 8 Producer 1 put: 9 Consumer 1 got: 9 |

Figure-5.25: Result of running ProducerConsumerSynchronized application

### **Summary**

In this exercise, you have learned how to perform inter-thread commmunication by the usage of wait(), notify(), and notifyAll() methods.