CSE2008 (Operating Systems) Lab – 5

KHAN MOHD OWAIS RAZA
20BCD7138

Q.1 Consider there are two producer and two consumer processes in the system. All processes share common buffer of size N. Producer processes produce an item and place it into the buffer while consumer processes take an item from the buffer and consume. All producer and consumer processes can be executed concurrently. Write a java program to demonstrate the aforementioned scenario using multithreading where each thread represents a producer or consumer process.

```
/* KHAN MOHD OWAIS RAZA */
/* 20BCD7138 */
/* CSE2008 (Operating Systems) Lab Practical */
/* Consider there are two producer and two consumer processes
   in the system. All processes share common buffer of size N.
   Producer processes produce an item and place it into the buffer &
   consumer processes takes item from the buffer and consume.
   All producer & consumer processes can be executed concurrently.
   Write a java program to demonstrate the scenario using
multithreading
   where each thread represents a producer or consumer process.
 */
package owaisraza_20bcd7138;
public class MyClass {
public static void main(String[] args) {
PRODUCER_CONSUMER_TEST C = new PRODUCER_CONSUMER_TEST();
PRODUCER P1 = new PRODUCER(C, 1);
CONSUMER C1 = new CONSUMER(C, 1);
PRODUCER P2 = new PRODUCER(C, 2);
CONSUMER C2 = new CONSUMER(C, 2);
P1.start();
C1.start();
P2.start();
C2.start();
}}
class PRODUCER_CONSUMER_TEST {
private int CONTENTS;
private boolean AVAILABLE = false;
public synchronized int get() {
while (AVAILABLE == false) {
try {
wait();
}
catch (InterruptedException e) {
}}
```

```
AVAILABLE = false;
notifyAll();
return CONTENTS;
}
public synchronized void put(int value) {
while (AVAILABLE == true) {
try {
wait();
}
catch (InterruptedException e) {
}}
CONTENTS = value;
AVAILABLE = true;
notifyAll();
}}
class CONSUMER extends Thread {
private PRODUCER_CONSUMER_TEST PRODUCER_CONSUMER_TEST;
private int NUMERICAL_VALUE;
public CONSUMER(PRODUCER_CONSUMER_TEST c, int NUMERICAL_VALUE) {
PRODUCER_CONSUMER_TEST = c;
this.NUMERICAL_VALUE = NUMERICAL_VALUE;
}
public void run() {
int value = 0;
for (int i = 0; i < 10; i++) {
value = PRODUCER_CONSUMER_TEST.get();
System.out.println("CONSUMER-" + this.NUMERICAL_VALUE + " GOT: " +
value);
}}
}
class PRODUCER extends Thread {
private PRODUCER_CONSUMER_TEST PRODUCER_CONSUMER_TEST;
private int NUMERICAL_VALUE;
public PRODUCER(PRODUCER_CONSUMER_TEST c, int NUMERICAL_VALUE) {
PRODUCER_CONSUMER_TEST = c;
this.NUMERICAL_VALUE = NUMERICAL_VALUE;
}
public void run() {
for (int i = 0; i < 10; i++) {
PRODUCER_CONSUMER_TEST.put(i);
System.out.println("PRODUCER-" + this.NUMERICAL_VALUE + " PUT: " +
i);
try {
sleep((int)(Math.random() * 100));
}
catch (InterruptedException e) {
}}
}}
```

Output :-

```
<terminated> MyClass [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe
PRODUCER-1 PUT: 0
CONSUMER-1 GOT: 0
CONSUMER-1 GOT: 0
PRODUCER-2 PUT: 0
CONSUMER-1 GOT: 1
PRODUCER-1 PUT: 1
CONSUMER-2 GOT: 1
PRODUCER-2 PUT: 1
CONSUMER-1 GOT: 2
PRODUCER-1 PUT: 2
PRODUCER-2 PUT: 2
CONSUMER-2 GOT: 2
CONSUMER-1 GOT: 3
PRODUCER-2 PUT: 3
PRODUCER-1 PUT: 3
CONSUMER-2 GOT: 3
PRODUCER-2 PUT: 4
CONSUMER-1 GOT: 4
CONSUMER-2 GOT: 4
PRODUCER-1 PUT: 4
PRODUCER-2 PUT: 5
CONSUMER-2 GOT: 5
PRODUCER-2 PUT: 6
CONSUMER-1 GOT: 6
CONSUMER-2 GOT: 5
PRODUCER-1 PUT: 5
CONSUMER-1 GOT: 7
PRODUCER-2 PUT: 7
PRODUCER-1 PUT: 6
CONSUMER-2 GOT: 6
CONSUMER-1 GOT: 8
PRODUCER-2 PUT: 8
CONSUMER-2 GOT: 9
PRODUCER-2 PUT: 9
CONSUMER-1 GOT: 7
PRODUCER-1 PUT: 7
CONSUMER-2 GOT: 8
PRODUCER-1 PUT: 8
PRODUCER-1 PUT: 9
CONSUMER-2 GOT: 9
```