



LAB - 8

Java Programming Lab

Topics Covered

Multithreading

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 8.1

Aim:

Write a java program to show multithreaded producer and consumer application.

Theory:

Class : A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

Threads : Threads allows a program to operate more efficiently by doing multiple things at the same time. Threads can be used to perform complicated tasks in the background without interrupting the main program.

Multithreading : Multithreading in Java is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc.

Wait() method : The wait() method is defined in object class which is the super most class in Java. This method tells the calling thread (Current thread) to give up the lock and go to sleep until some other thread enters the same monitor and calls notify() or notifyAll(). It is a final method , so we can't override it.

java.lang.Object.notifyAll() : It wakes up all threads that are waiting on this object's monitor. A thread waits on an object's monitor by calling one of the wait methods. The awakened threads will not be able to proceed until the current thread relinquishes the lock on this object. The awakened threads will compete in the usual manner with any other threads that might be actively competing to synchronize on this object; for example, the awakened threads enjoy no reliable privilege or disadvantage in being

the next thread to lock this object. This method should only be called by a thread that is the owner of this object's monitor.

Thread.sleep() : Thread.sleep() method can be used to pause the execution of current thread for specified time in milliseconds. The argument value for milliseconds can't be negative, else it throws IllegalArgumentException

Source Code:

```
package javaapplication1;

/**
 * @author reeha
 */

import java.lang.*;

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

class Threadsys {
    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 Threadsys threadsys;
    private int number;
```

```
public Consumer(Threadsys c, int number) {
    threadsys = c;
    this.number = number;
}

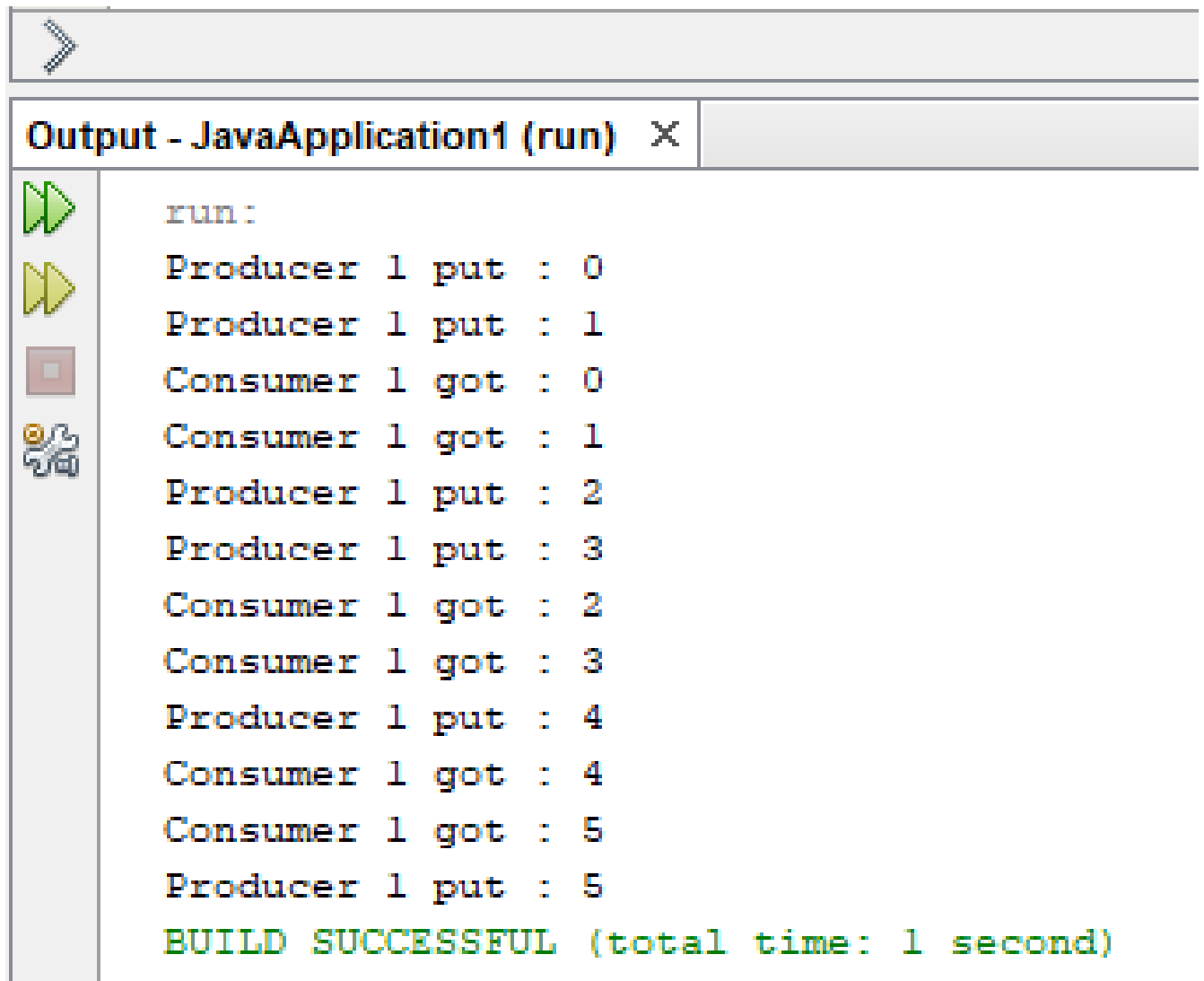
public void run() {
    int value = 0;
    for (int i = 0; i <= 5; i++) {
        value = threadsys.get();
        System.out.println("Consumer " + this.number + " got : " + value);
    }
}

}

class Producer extends Thread {
    private Threadsys Threadsys;
    private int number;

    public Producer(Threadsys c, int number) {
        Threadsys = c;
        this.number = number;
    }

    public void run() {
        for (int i = 0; i <= 5; i++) {
            Threadsys.put(i);
            System.out.println("Producer " + this.number + " put : " + i);
        }
    }
}
```

Output:

The image shows a screenshot of an IDE's output window. At the top is a toolbar with a right-pointing arrow icon. Below it is a tab labeled "Output - JavaApplication1 (run)" with a close button (X). The main area of the window displays the output of a Java program. On the left side of the output area is a vertical toolbar with four icons: a green double arrow (run), a yellow double arrow (debug), a red square (stop), and a gear icon (configuration). The output text is as follows:

```
run:
Producer 1 put : 0
Producer 1 put : 1
Consumer 1 got : 0
Consumer 1 got : 1
Producer 1 put : 2
Producer 1 put : 3
Consumer 1 got : 2
Consumer 1 got : 3
Producer 1 put : 4
Consumer 1 got : 4
Consumer 1 got : 5
Producer 1 put : 5
BUILD SUCCESSFUL (total time: 1 second)
```

EXPERIMENT – 8.2

Aim:

Write an application that executes two threads. One thread displays "A" every 1000 milliseconds and other displays "B" every 3000 milliseconds. Create the threads by extending the Thread class.

Theory:

Class : A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

Threads : Threads allows a program to operate more efficiently by doing multiple things at the same time. Threads can be used to perform complicated tasks in the background without interrupting the main program.

Multithreading : Multithreading in Java is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc.

Thread.sleep() : Thread.sleep() method can be used to pause the execution of current thread for specified time in milliseconds. The argument value for milliseconds can't be negative, else it throws IllegalArgumentException

Source Code:

```
package javaapplication1;

/**
 *
 * @author reeha
 */

class A extends Thread {

    public void run() {
        for (;;) {
            System.out.println("A");
            try {
                Thread.currentThread().sleep(1000);
            }

            catch (InterruptedException e) {
            }
        }
    }
}

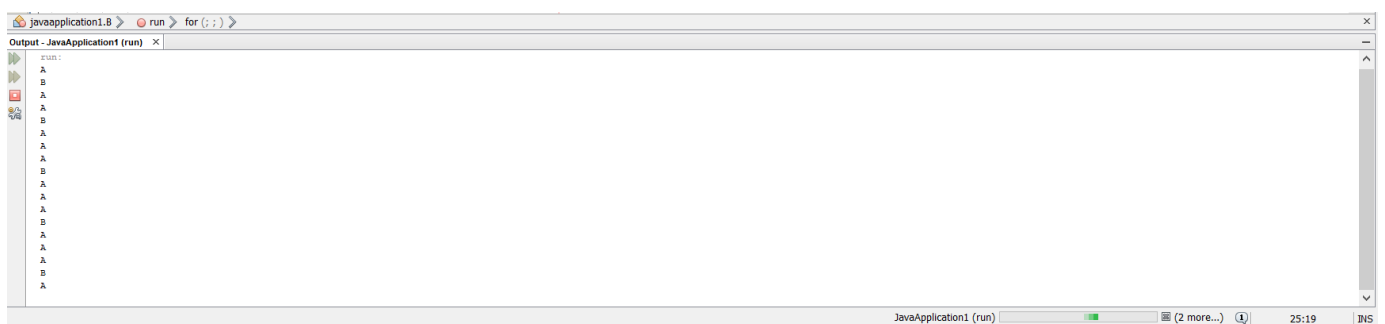
class B extends Thread {
    public void run() {
        for (;;) {
            System.out.println("B");
            try {
                Thread.currentThread().sleep(3000);
            }
        }
    }
}
```

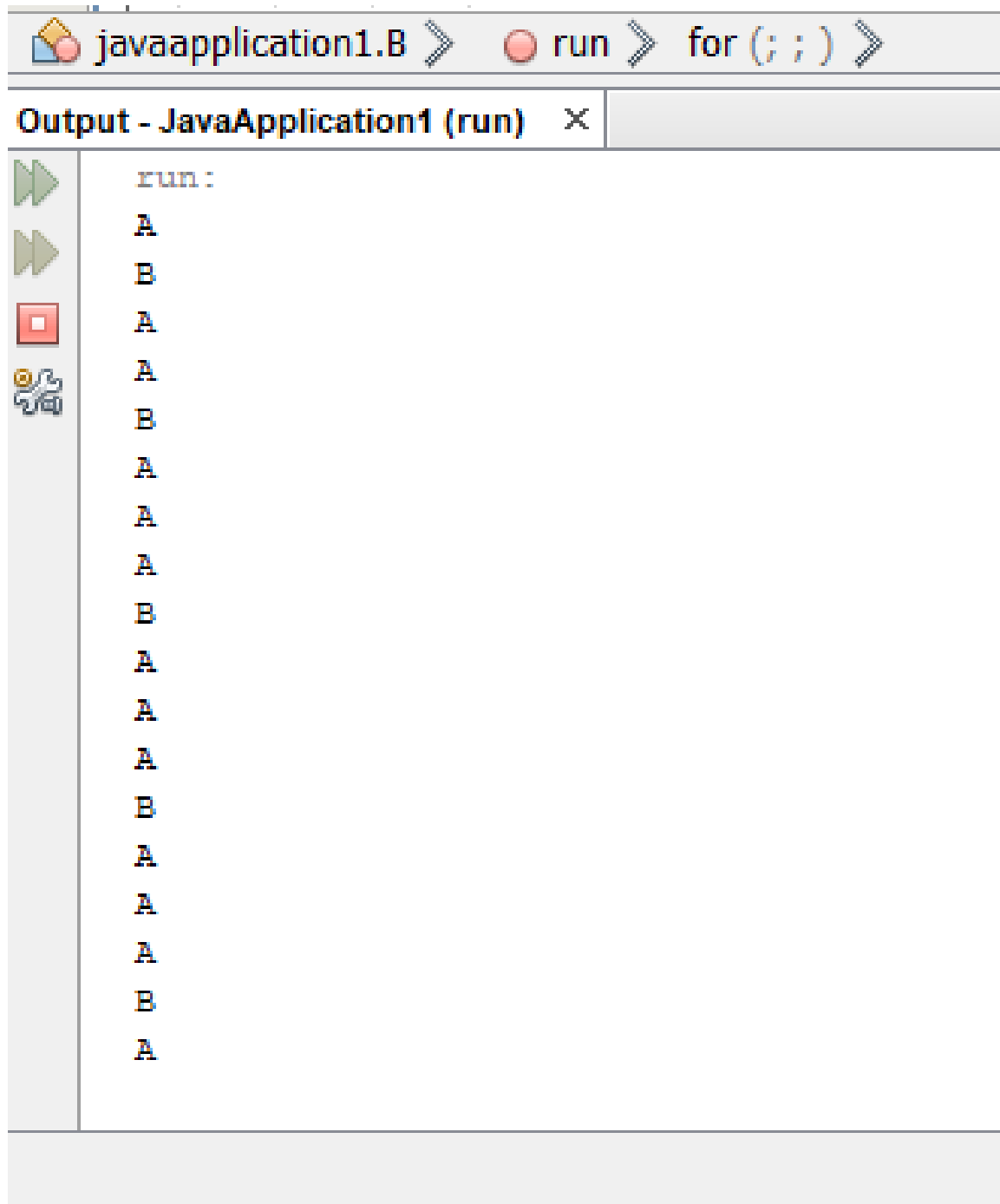


```
        } catch (InterruptedException e) {  
        }  
    }  
}  
}
```

```
public class threading {  
    public static void main(String args[]) {  
  
        A obj1 = new A();  
        B obj2 = new B();  
        obj1.start();  
        obj2.start();  
    }  
}
```

Output:





```
run:
A
B
A
A
B
A
A
B
A
A
B
A
A
B
A
```

Viva Questions

1. What is multithreading?

Ans.

Multithreading is a process of executing multiple threads simultaneously. Multithreading is used to obtain the multitasking. It consumes less memory and gives the fast and efficient performance. Its main advantages are:

- Threads share the same address space.
- The thread is lightweight.
- The cost of communication between the processes is low.

2. Differentiate between process and thread?

Ans.

There are the following differences between the process and thread.

A Program in the execution is called the process whereas; A thread is a subset of the process.

Processes are independent whereas threads are the subset of process.

Process have different address space in memory, while threads contain a shared address space.

Context switching is faster between the threads as compared to processes.

3. What is the purpose of wait() method in Java?

Ans.

The wait() method is provided by the Object class in Java. This method is used for inter-thread communication in Java. The java.lang.Object.wait() is used to pause the current thread, and wait until another thread does not call the notify() or notifyAll() method. Its syntax is given below.

```
public final void wait()
```

4. Explain different way of using thread?

Ans.

The thread could be implemented by using runnable interface or by inheriting from the Thread class. The former is more advantageous, 'cause when you are going for multiple inheritance..the only interface can help.

5. Describe synchronization in respect to multithreading.

Ans.

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared variable while another thread is in the process of using or updating same shared variable. This usually leads to significant errors.

6. True or false: An applet can run multiple threads.

Ans.

True. The paint and update methods are always called from the AWT drawing and event handling thread. You can have your applet create additional threads, which is recommended for performing time-consuming tasks.