CORE JAVA

MANUAL V8.3

MODULE CODE:

ANUDIP FOUNDATION


ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

Module 6: Thread and Exception Handling

Chapter 3

Objective: After completing this lesson you will be able to : * Gain an understanding of inter-thread communication and the join(), sleep(), wait(), notify() methods * Learn about Thread Synchronization and Deadlock in Java	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 60 minutes	**Practical Duration:** 60 minutes
Total Duration: 120 minutes	

Chapter 3

3.1 Inter-thread Communication-join(), sleep(), wait(), notify()

Java inter-thread communication or cooperation deals with enabling communication between synchronized process threads. In cooperation, one thread gets paused while running in its code block (critical section), while another enters the code block for execution.

Object class methods used for enabling inter-thread communication are -

i) notify()
ii) wait()
iv) join()
v) sleep()

How the methods to enable and perform inter-thread communication -

i) notify() - It wakes a solitary thread present on an object monitor. A waiting thread is selected and triggered randomly for execution.

Syntax: public final void notify()

iii) wait() - It unlocks a current thread and waits for another to be initiated through a notify() or notifyAll() method, or after the passing of a designated time.

The two wait() methods are -

* public final void wait()throws InterruptedException - waits for object notification

* public final void wait(long timeout)throws InterruptedException - waits for a designated time

iii) join() - It is a method that puts a currently running thread on wait until the thread called on it is eliminated.

Syntax: public final void join()

iv) sleep() – This method is used for assigning a sleep state to a thread for a designated amount of time.

The two sleep() methods are –

* public static void sleep(milliseconds length)throws InterruptedException
* public static void sleep(milliseconds length, int nanos)throws InterruptedException

Code example of inter-thread communication –

```
class User{
int amount=15000;
synchronized void withdrawal(int amount){
System.out.println('will withdraw');
if(this.amount<amount){
System.out.println('Low balance; waiting for deposit');
try{wait();}catch(Exception e){}
}
this.amount-=amount;
System.out.println('withdrawal completed');
  }
synchronized void deposit(int amount){
System.out.println( "to be deposited');
this.amount+=amount;
System.out.println('depositing completed');
notify();
}
  }
class Test{
public static void main(String args[]){
final User U=new User();
new Thread(){
public void run(){u.withdrawal(20000);}
```

```
}.start();
new Thread(){
public void run(){u.deposit(15000);}
}.start();
}}
```

Output:

```
will withdraw
Low balance; waiting for deposit
to be deposited
depositing completed
withdrawal completed
```

3.2 Thread Synchronization and Deadlock

Java thread synchronization refers to controlling the accessibility of multiple threads to a shared resource. Using synchronization is ideal for allowing only one thread to access a shared resource. Synchronization is used to reduce thread interference and to maintain stability.

i) Thread Synchronization

Java thread synchronization types are -

* Mutually exclusive
* Inter-thread communication

Example code of thread synchronization -

```
class Table{
synchronized void printTable(int n){
for(int i=1;i<=5;i++){
System.out.println(n*i);
try{
Thread.sleep(400);
  }catch(Exception e){System.out.println(e);}
}
}
}
```

```
public class TestSynchronization3{
public static void main(String args[]){
final Table obj = new Table();
Thread t1=new Thread(){
public void run(){
obj.printTable(5);
}
};
Thread t2=new Thread(){
public void run(){
obj.printTable(100);
}
};
t1.start();
t2.start();
  }
}
```

Output:

```
5
10
15
20
25
100
200
300
400
500
```

ii) Deadlock

In Java, deadlock is a situation where multiple threads are blocked, waiting on one another. It occurs when two or more threads require the same locks, but in varying order. Deadlock can happen if a synchronized keyword initiates

the current thread to block when it is waiting for a lock.

Code example of Java deadlock –

```
public class TestDeadlockDemo1 {
  public static void main(String[] args) {
    final String resource1 = 'karan kumar';
    final String resource2 = 'ratan kumar';
    Thread t1 = new Thread() {
      public void run() {
        synchronized (resource1) {
        System.out.println('Thread 1: resource 1 locked');
        try { Thread.sleep(200);} catch (Exception e) {}
        synchronized (resource2) {
          System.out.println('Thread 1: resource 1 locked');
        }
      }
    }
    };

     Thread t2 = new Thread() {
    public void run() {
       synchronized (resource2) {
        System.out.println('Thread 2: resource 2 locked ');
        try { Thread.sleep(100);} catch (Exception e) {}
        synchronized (resource1) {
          System.out.println('Thread 2: resource 1 locked ');
        }
      }
    }
    };
    t1.start();
    t2.start();
  }
}
```

Output:

Thread 1: resource 1 locked

Thread 2: resource 2 locked

Practical (60 minutes)

See the example programme for Java thread synchronization below. Write the same programme with int i=2;i<=6,i++. Show the resulting output. Repeat the same programme for int=3;i<=9,i++.

```
class Table{
synchronized void printTable(int n){
for(int i=1;i<=5;i++){
System.out.println(n*i);
try{
Thread.sleep(400);
 }catch(Exception e){System.out.println(e);}
}
}
}
public class TestSynchronization3{
public static void main(String args[]){
final Table obj = new Table();
Thread t1=new Thread(){
public void run(){
obj.printTable(5);
```

```
}
};
Thread t2=new Thread(){
public void run(){
obj.printTable(100);
}
};
t1.start();
t2.start();
 }
}
```

Instructions: The progress of students will be assessed with the exercises mentioned below.

MCQ (10 minutes)

1. Java inter-thread communication is also known as _____.

a) composition

b) cooperation

c) collaboration

d) None of the mentioned

2. Inter-thread communication deals with enabling communication between _____ process threads

a) synchronized

b) asynchronized

c) dynamized

d) None of the mentioned

3. Where does a thread get paused?

a) binary data section

b) logical section

c) critical section

d) None of the mentioned

4. Which of these is not an inter-thread communication method?

a) sleep()

b) notify()

c) wait()

d) append()

5. notify() wakes _____ present on an object monitor.

a) two threads

b) multiple threads

c) a single thread

d) None of the mentioned

6. _____ unlocks a current thread and waits for another to be initiated through notify().

a) join()

b) wait()

c) notifyAll()

d) None of the mentioned

7. sleep() assigns a _____ state to a thread.

a) dormant

b) sleep

c) blocked

d) None of the mentioned

8. Java thread _____ refers to the controlling the accessibility of multiple threads.

a) assimilation

b) synchronization

c) initiation

d) None of the mentioned

9. Synchronization can reduce thread _____.

a) inheritance

b) interference

c) instanting

d) None of the mentioned

10. _____ threads are blocked by a Java deadlock.

a) Three

b) Single

c) Multiple

d) None of the mentioned