What is a Starvation ?

Starvation of thread in java is said to occur when a particular thread does not get access to the object or the resource which leads to an increase in waiting and execution time.

Starvation is said to occur when two or more threads are allocated to the CPU (Central Processing Unit) and takes a lot of time in execution, due to which other waiting threads cannot get the CPU for its execution to carry on.

**Causes Of Starvation:**

There are many reasons for causes of starvation of threads in java, some of them are described below:

**-1High Priority Running Thread:** There may be a case where a high priority thread is running by occupying the CPU and it needs heavy processing which requires a lot of time in completion, so for this work to get completely executed the other threads which have a low priority order have to wait for a long time which leads to starvation.

**-2Synchronized Block:** There may be a case where the order in which the threads are allowed to enter the **synchronized block** is granted the resources in the same order as they are programmed to be scheduled, which results in waiting for the resources and the objects by another thread leading to starvation, where the other threads other than a particular thread are given the CPU for its execution.

Example of Starvation:

Class MyThread extends Thread {

Public void run() {

String threadName = Thread.currentThread().getName();

System.out.println(threadName + “ Started”);

Synchronized(MyThread.class) { // lock

// doing some useful work

Try {

Thread.sleep(2000); // 2 sec

} catch (InterruptedException ie){}

}

System.out.println(threadName + “ End”);

}

}

Public class Test {

Public static void main(String[] args) {

System.out.println(“Start of Main thread”);

MyThread mt[] = new MyThread[10];

For (int i=0; i<mt.length; i++) { Mt[i] = new MyThread(); // create thread

Mt[i].start();

}

System.out.println(“End of Main thread”);

In this example, the main thread created 10 child threads. To execute some portion of the run() method (synchronized block) each child thread needs the lock of the current class. At a time only one thread can get the lock of one object. And to complete execution, each thread required more than 2 seconds time.

Among these 10 threads, there will be some threads executing at last. They were waiting for a long period of time because the thread was unable to gain regular access to shared resources (lock of current class) and was unable to make progress. In the above output, thread-8 waited for a long period of time

**Another Example Of Starvation:**

class Starvation extends Thread {   
static int count = 1;   
public void run() {   
System.out.println(count + " Thread execution starts");   
System.out.println("Thread execution completes");   
count++;   
}   
public static void main(String[] args) throws InterruptedException {   
System.out.println("Parent thread execution starts");   
  
/\* Priority of each thread given. \*/  
/\* Thread 1 with priority 7. \*/  
Starvation thread1 = new Starvation();   
thread1.setPriority(7);   
/\* Thread 2 with priority 6. \*/  
Starvation thread2 = new Starvation();   
thread2.setPriority(6);   
/\* Thread 3 with priority 5. \*/  
Starvation thread3 = new Starvation();   
thread3.setPriority(5);  
/\* Thread 4 with priority 4. \*/  
Starvation thread4 = new Starvation();   
thread4.setPriority(4);   
/\* Thread 5 with priority 3. \*/  
Starvation thread5 = new Starvation();   
thread5.setPriority(3);   
  
thread1.run();   
thread2.run();   
thread3.run();   
thread4.run();   
thread5.run();   
  
System.out.println("Parent thread execution completes");   
}   
}

Here thread 5 have to wait because of the other threads

Here thread 5 have to wait because of the other threads have high. **Priority more than thread 5 .**

How did solve Stavation:

Some of the important points to remove starvation of threads are given as follows:

1-By implementation of the thread.yeid()  method, so that when the thread in the process after releasing the lock gets a fair chance to occupy the C.P.U. and can get some time to complete its execution till the original thread again gets the control over the C.P.U.

2-One can also use the thread.sleep()   method to given chance to other Threads for execution. (As We Used)