Course Name:

Advanced Java

Lecture 7
Topics to be covered

- Multithreading

Thread-An Introduction

- A thread is defined as the path of execution of a program.
- It is a sequence of instructions that is executed to define a unique flow of control.
- It is the smallest unit of code.
- <u>Example:- </u>A central processing unit performs various tasks simultaneously, such as writing and printing a document, installing a software and displaying the date and time on the status bar. All these processes are handled by separate threads.

Cont…

- A process that is made up of only one thread is known as single-threaded application.
- A process that creates two or more threads is called a multi threaded application. For example:- Any web browser, such as Internet Explorer is a multithreaded application.
- Each thread in a multithreaded program runs at the same time and has a different execution path.

Basic Concept of Multithreading

- A single threaded application can perform only one task at a time. You have to wait for one task to complete before another can start.
- Threads are used when you have to run various applications that perform large and complex computations. Multithreading helps to perform these operations simultaneously, saving the time of the user.
- Every program has at least one thread and you can create more threads when necessary.

Cont..

- The microprocessor allocates memory to the processes that you execute . Each process occupies its own address space or memory.
- However, all threads in a process occupy the same address space.

Multitasking

- Multitasking is the ability to execute more than one task at the same time.
- Multitasking can be divided into the following categories:

1. Process-based Multitasking
2. Thread-based Multitasking

Process-Based Multitasking

- A process is a program that is being executed by the processor.
- The process-based multitasking feature of java enables you to switch from one program to another so quickly that it appears as if the programs are executing at the same time.
- Example:- Process-based multitasking enables you to run the java compiler and use the text editor at the same time.
- This feature enables a computer to execute two or more processes concurrently.
- Processes are the tasks that require separate address spaces in the computer memory.

Thread-based Multitasking

- A single program can contain two or more threads and therefore, perform two or more tasks simultaneously.

- Example:- A text editor can perform writing to a file and print a document simultaneously with separate threads performing the writing and printing actions.
- Threads are called lightweight processes because there are fewer overloads when the processor switches from one thread to another. On the other hand, when the processor switches from one process to another process the overload increases.

Benefits of Multithreading

- Improved performance:- Provides improvement in the performance of the processor by simultaneous execution of computation and input/output operations.
- Minimized system resource usage:- Minimizes the use of system resources by using threads, which share the same address space and belong to the same process.
- Simultaneous access to multiple applications:- Provides access to multiple applications at the same time because of quick context switching among threads.
- Program structure simplification:- Simplifies the structure of complex applications, such as multimedia applications. Sub programs can be written for each activity that makes complex

Pitfalls of Multithreading

- Race Condition:- When two or more threads simultaneously access the same variable, at least one thread tries to write a value in this variable. This is called race condition. This condition is caused by the lack of synchronization between two threads.

Example:- In a word processor, there are two threads, one to read a file and the other to write before performing its operation. The race condition arises when the thread to read a file, reads the file, before the thread to write a file performs its operation

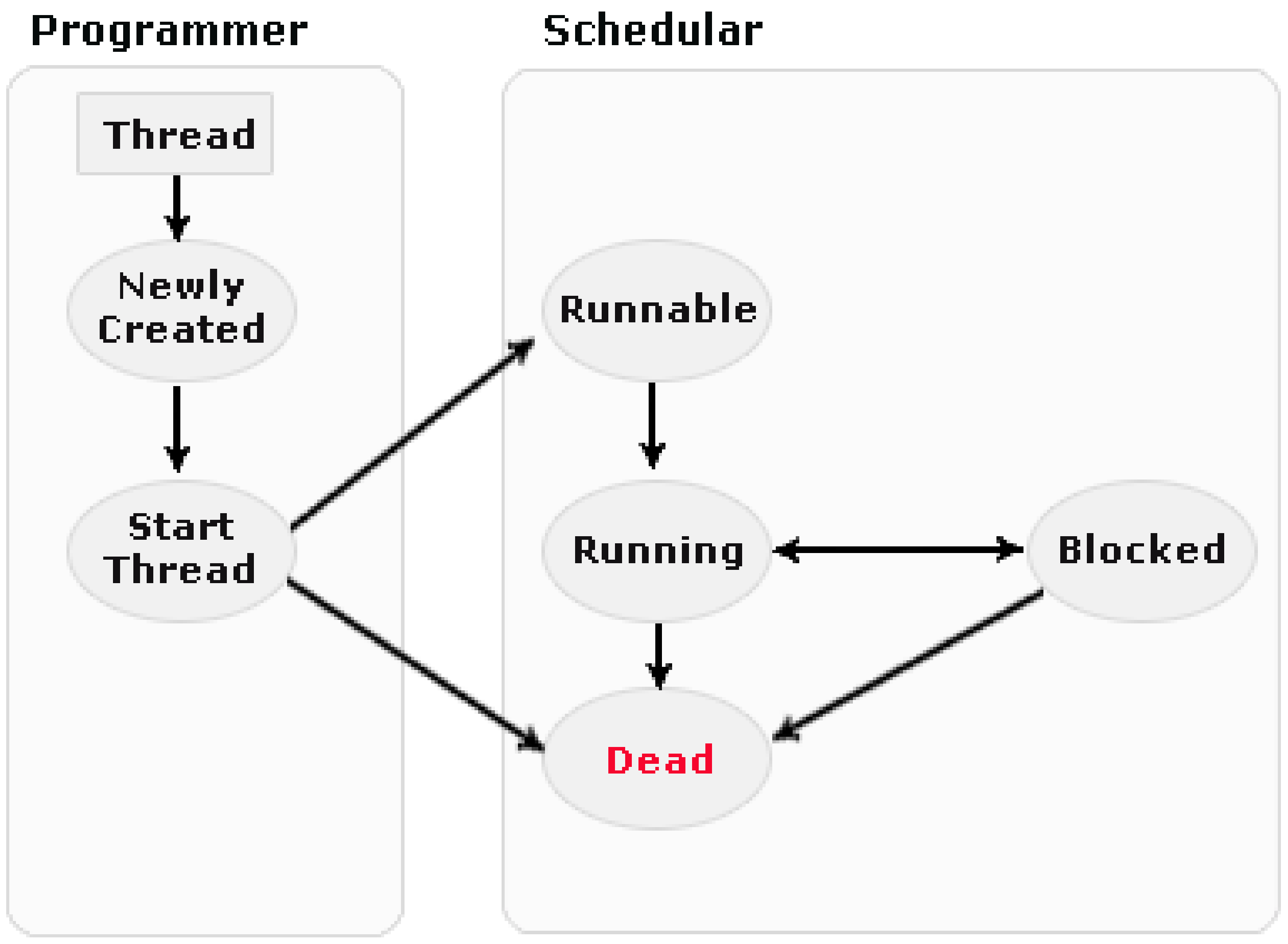
Cont…

- Deadlock condition:- This condition arises in a computer system when two threads wait for each other to complete their operations before performing their individual actions. As a result the two threads become locked and the program fails.

Example:- Consider two threads, Thread A and Thread B. Thread A is waiting for a lock to be released by Thread B, and Thread B is waiting for a lock to be released by Thread A to complete its transaction.

- Lock Starvation:- It occurs when the execution of a thread is postponed because of its low priority. The java run time environment executes threads based on their priority because the CPU can execute only one thread at a time.

Life Cycle of Thread



Cont…

- A thread can be in any of the following states at any instant of time.
- New:- When a thread object is created, a thread is born. This state is called a new state. At this stage, The new thread can start and move to the Runnable state or this thread can be killed immediately. Then it is called Dead thread.
- Runnable:- It means that the thread is ready for execution and is waiting for the availability of the processor for execution. This stage comes between new and running state.

Cont…

- Blocked:- A thread is in a blocked state when it is not allowed to enter the runnable state . This can happen when a thread is either suspended or sleeping. At this stage, thread is not considered dead.
- Waiting:- This is the state when a thread is waiting due to round robin scheduling of the processor.
- Terminated:- A thread is considered terminated when it has completed its execution.

Thread Implementation

- There are two ways of implementing threading in java:-
  - By extending java.lang.Thread class, or
  - By implementing java.lang.Runnable interface
- Implementing Runnable is better because in Java we can only extend one class so if we extend Thread class we can not extend any other class while by implementing Runnable interface we still have that option open with us.
- Second reason which make sense to me is more on OOPS concept according to OOPS if we extend a class we provide some new feature or functionality , So if the purpose is just to use the run() method to define code its better to use Runnable interface.

Multithreading using Runnable Interface

- The Runnable interface only consists of run() method , which is executed when the thread is activated.
- Syntax to declare run() method is:

  Public void run()

  The run() method contains the code that defines the new thread.

Example of multithreading using Runnable Interface

```
class newThreadClass implements Runnable
{
    String ThreadName;
    newThreadClass(String name)
    {
            ThreadName=name;
            Thread t=new Thread(this,ThreadName);
            System.out.println("Thread created: "+t);
            t.start();
    }
public void run()
    {
            try
            {
                    for(int i=1;i<=5;i++)
                    {
                            System.out.println(ThreadName + "loop:" +i);
                            Thread.sleep(100);
                    }
            }
            catch(InterruptedException obj)
            {
                    System.out.println("Thread:" + ThreadName +
                                                    "interrupted");
            }
            System.out.println(ThreadName + "is existing");
    }
}
```

```
class MultipleThread1
{
    public static void main(String args[])
    {
            new newThreadClass("FirstChildThread");
            new newThreadClass("SecondChildThread");
            try{
                    for(int i=1;i<=5;i++){
                            System.out.println("main Thread loop:" +i);
                            Thread.sleep(300);
                    }
            }
            catch(InterruptedException obj)
            {
                    System.out.println("Main thread is interrupted");
            }
            System.out.println("main thread is terminating now");
    }
}
```

Multithreading using Thread class

- You can create threads by extending the Thread class.
- The Thread class is defined in the java.lang package. This class defines several methods that can be overridden by a derived class.
- You can use the run() method to create threads only if a class does not extend to any other class. The extending class calls the start() method to begin the child thread execution.

Example of Multithreading using Thread class

```
class ThreadDemo extends Thread
{
    ThreadDemo()
    {
            super("ChildThread");
            System.out.println("ChildThread:"+this);
            start();
    }
    public void run()
    {
            System.out.println("The child thread started");
            System.out.println("Exiting the child thread");
    }
}
```

```
class ThreadDemoClass
{
    public static void main(String args[])
    {
            new ThreadDemo();
            System.out.println("The main thread started");
            System.out.println("The main thread sleeping");
            try
            {
                        Thread.sleep(1000);
            }
            catch(InterruptedException e)
            {
                        System.out.println("The main thread interrupted");
            }
            System.out.println("Exiting main thread");
    }
}
```