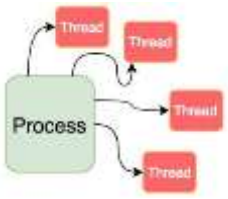
Unit – 06: Multithreaded Programming and I/O

- The java thread model, thread priorities, synchronization, creating a thread, creating multiple thread,
- using is Alive() and join(),
- Synchronization in multiple thread,
- I/O basics, streams(byte and character),
- Reading and writing console input and output, Reading and writing files.

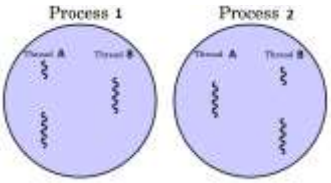
Questions to be discussed:

1. What is thread? Explain life cycle of thread in Java.
2. Differentiate between multi-tasking and multi-threading.
3. What do you mean by resuming and stoping of threads? Explain it.
4. What do you mean by stream in Java? Discuss its type in brief.
5. Write the difference between character stream and byte stream.
6. Explain input and output console in Java.
7. Write short notes on:
   a. Thread synchronization
   b. Alive() and join()

What is a Thread in Java?



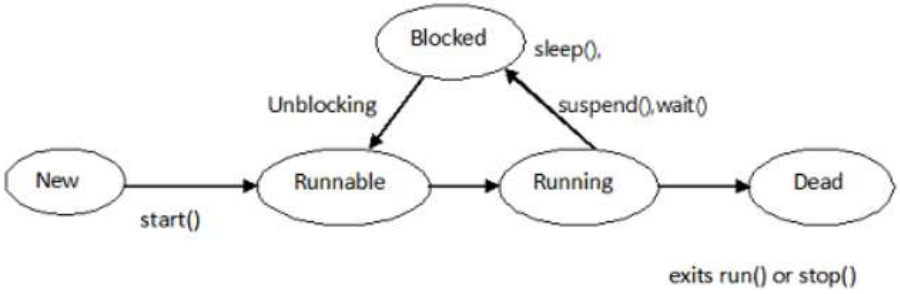
- Thread is a sub part of process.
- It is also known as light weight process.
- Multiple thread within a process share process state memory etc.
- Each thread in the program has its own program counter, stack, and local variable.
- All the programs have at least one thread, known as main thread, that is provided by the JVM at the starting of the program's execution.
- Multiple threads of execution can be run concurrently on the Java Virtual Machine.
- The priority of each thread varies, higher priority threads are executed first.



Lifecycle of a Thread in Java:

There are basically 5 stages in the lifecycle of a thread, as given below:

1. New
2. Runnable
3. Running
4. Blocked
5. Dead



New:

- A new thread is created but not working.
- A thread after creation and before invocation of start() method will be in new state.

Runnable:

- A thread after invocation of start() method will be in runnable state.
- A thread in runnable state will be available for thread scheduler.

Running:

- A thread in execution after thread scheduler select it, it will be in running state.

Blocked:

- A thread which is alive but not in runnable or running state will be in blocked state.
- A thread can be in blocked state because of suspend(), sleep(), wait() methods or implicitly by JVM to perform I/O operations.

Dead:

- A thread after exiting from run() method will be in dead state.
- We can use stop() method to forcefully killed a thread.

Multithreading in Java:

- In Java, multithreading is the method of running two or more threads at the same time to maximize CPU utilization.
- As a result, it is often referred to as Concurrency in Java.
- Each thread runs in parallel with the others.
- Since several threads do not assign different memory areas, they conserve memory.
- Furthermore, switching between threads takes less time.
- In Java, multithreading enhances program structure by making it simpler and easier to navigate.

Difference between Multi-tasking and Multi-threading:

Multitasking	Multithreading
In multitasking, users are allowed to perform many tasks by CPU.	In multithreading, many threads are created from a process.
Here, the processes share separate memory.	Here, processes are allocated the same memory.
It involves in multiprocessing.	It does not involve in multiprocessing.
In multitasking, the CPU is provided in order to execute many tasks at a time.	In multithreading, a CPU is provided in order to execute many threads of a process at a time.
In multitasking, processes don't share the same resources.	While in multithreading, each process shares the same resources.
Multitasking is slow compared to multithreading.	While multithreading is faster.
It helps in developing efficient programs.	It helps in developing efficient operating systems.

Java Thread Priorities:

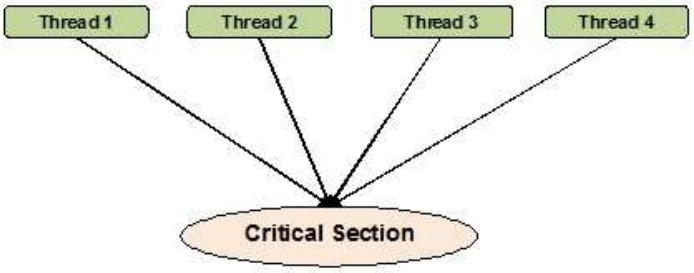
- The number of services assigned to a given thread is referred to as its priority.
- Any thread generated in the JVM is given a priority.
- The priority scale runs from 1 to 10.
- 1 is known as the lowest priority and 10 represents the highest level of priority.
- The main thread's priority is set to 5 by default, and each child thread will have the same priority as its parent thread.

Synchronization in multiple thread:

- Synchronization is a process of handling resource accessibility by multiple thread requests.
- The main purpose of synchronization is to avoid thread interference.
- At times when more than one thread try to access a shared resource, we need to ensure that resource will be used by only one thread at a time.
- The process by which this is achieved is called synchronization.
- The synchronization keyword in java creates a block of code referred to as critical section.

Syntax:

```
synchronized (object)
{
  //statement to be synchronized
}
```



Java Thread class:

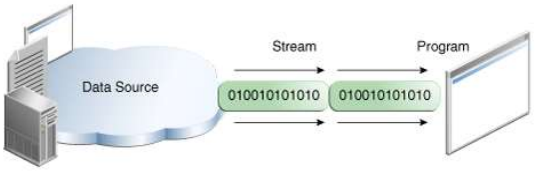
- Java provides **Thread class** to achieve thread programming.
- Thread class provides methods to create and perform operations on a thread.
- Some important thread methods are:
  - start()
  - stop()
  - run()
  - suspend()
  - resume() etc.

Method	Description
start()	It is used to start the execution of the thread.
run()	It is used to do an action for a thread.
isAlive()	It tests if the thread is alive.
yield()	It causes the currently executing thread to pause & allow other threads to execute temporarily.
suspend()	It is used to suspend the thread.
resume()	It is used to resume the suspended thread.
stop()	It is used to stop the thread.

isAlive() and join():

- Sometimes one thread needs to know when other thread is terminating.
- In java, **isAlive()** and **join()** are two different methods that are used to check whether a thread has finished its execution or not.
- The **isAlive()** method returns **true** if the thread is still running otherwise it returns **false**.
- But, **join()** method is used more commonly than **isAlive()**.
- This method waits until the thread on which it is called terminates.
- Using **join()** method, we tell our thread to wait until the specified thread completes its execution.
- If we run a thread without using join() method then the execution of thread cannot be predict.

What is Stream?



- A stream is a sequence of data.
- In Java, a stream is composed of bytes.
- Java programs perform I/O operations using streams.
- It's called a stream because it is like a stream of water that continues to flow.
- An **input stream** is used to read data from the source.
- An **output stream** is used to write data to the destination.

Types of Streams:

Depending upon the data a stream can be classified into two types:

1. Byte Stream
2. Character Stream

Difference between character stream and byte stream:

Character stream	Byte stream
Character stream is used to read and write a single character of data.	Byte stream is used to read and write a single byte (8 bits) of data.
Reading or writing a character or text based I/O such as text file, text documents, XML, HTML etc.	Reading or writing to binary data such as exe file, image file, LLL formats file like .zip, .class, .exe etc.
Input and output character streams are called readers and writers respectively.	Input and output byte streams are simply called input streams and output streams respectively.
The abstract class reader and writer and their derived classes in java.io package provide support for character streams.	The abstract class InputStream and OutputStream and their derived classes in java.io package provide support for byte streams.

What is a Console?

- To run a program, we might need input from the user according to the requirement.
- We cannot always take input just from the program.
- Sometimes, we can take the input from the console or terminal too.
- The process of taking input from the console is introduced by the concept of Java Console.
- Java programming language provides several ways in order to take input from the console and provide its corresponding output on the same console.
- There are three ways to take the input from the Java console. They are:
  1. Using Java Scanner class (Basic level)
  2. Using Java BufferedReader class (Intermediate level)
  3. Using Java Console class

Scanner class in Java:

- A class that is used to scan inputs within the program is known as the Scanner class.

BufferedReader class in Java:

- It is one of the classical methods of taking input from the user.
- A new statement, " InputStreamReader " is also introduced along with the " BufferedReader " in order to scan the input values using BufferedReader.

Java Console class:

- The class " Console " in Java was introduced from Java version 1.5.
- The class " Console " can be accessed through the package " Java.io " which is the basic package that is used in all programs constructed in Java.
- There are several methods that are embedded within the " Console " class.

Methods in Console class:

readLine()

- The method that is used to read a single line of text or String from the console is the " readLine() " method.
- This method can be accessed or used when an object is created within the String class.

readPassword()

- The method that is used to read or scan a password as an input in a way such that the input taken is shown or visible at any time on the screen is the " readPassword() " method.
- **reader()**

- The method is called by an object created in the class " Reader() " which handles and relates to the console.
- This method also helps in the creation of the object in the class " Reader() ".

Console printf(String, Object)

- The method that is used in order to print a string within the output of the console is " printf() ".
- It can be accessed using the class " Console() ".

flush()

- The method that is used to clear all the statements or the matter present on the console is " flush() ".
- It is a general method which is neither creates an object nor will be accessed by an object created within a particular class.
- It just removes the matter on the console screen. Its return data type is generally " void ".