CSE 1203

Object Oriented Programming

Java Special Topics

Singleton Class

Java: Singleton Class

Java: swing

A **Singleton** class is a class that allows only one instance of itself to be created and provides a global point of access to that instance. This is achieved by making the constructor private, so that no other instances of the class can be created, and providing a static method that returns the single instance of the class.
The Singleton pattern is commonly used in situations where a single instance of a class needs to coordinate actions across the system, such as in a logging or configuration system. It ensures that there is only one instance of the class, which can be accessed globally, preventing unnecessary duplication and ensuring consistency across the system.

Implementing a Singleton Class

1.Declare a private static variable to hold the single instance of the class.
2.Make the constructor of the class private, so that no other instances can be created.
3.Provide a public static method to return the single instance of the class, creating it if necessary.

Java: Singleton Class Implementation

```
class Singleton {
    private static Singleton instance=null;
    private Singleton() {}
    public static  Singleton getInstance()
{
         if (instance == null)
          instance = new Singleton();
      return instance;
}
public void Show(){
    System.out.println("Sigleton show");
}
}
public class MyFirst {
 public static void main(String[] args) {

  Singleton single1=Singleton.getInstance();
  single1.Show();
  Singleton single2=Singleton.getInstance();
  if(single1==single2)
    System.out.println("Both are equal");
    }
}
```

Multithreading

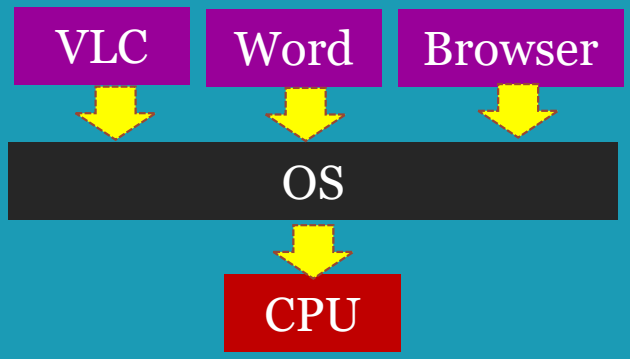
Java: Multitasking, Multiprocessing & Multithreading

Multitasking

Multitasking, the running of multiple programs (sets of instructions) in one computer at the same time. Multitasking is used to keep all of a computer's resources at work as much of the time as possible

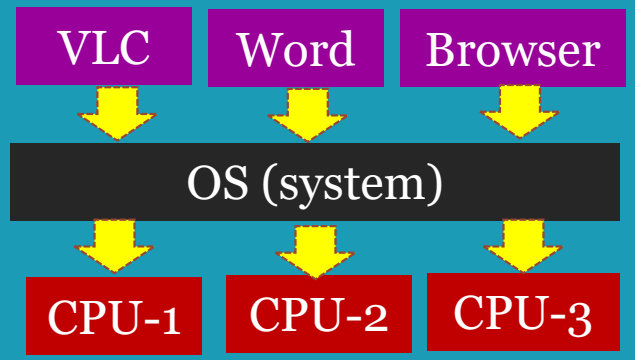
Real-life cxamples include chewing gum while walking, sending e-mails during a meeting, and talking on the phone while watching television.

Example: Computer OS can executes several tasks simultaneously.



Types of Multitasking

1. **Processed based Multitasking (Multiprocessing)**
   When one system connected to several CPUs in order to complete the task. Suitable of system level.



Types of Multitasking

2. Thread based Multitasking (Multithreading)

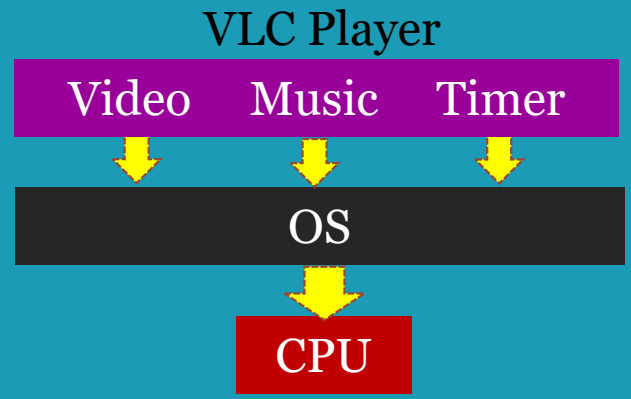
Process/task/Job: .a complete task

Thread: sub-tasks in a process

Ex: VLC is a process and in VLC Video, Music, Timer etc. are threads

Multithreading: performs multiple threads in one time.

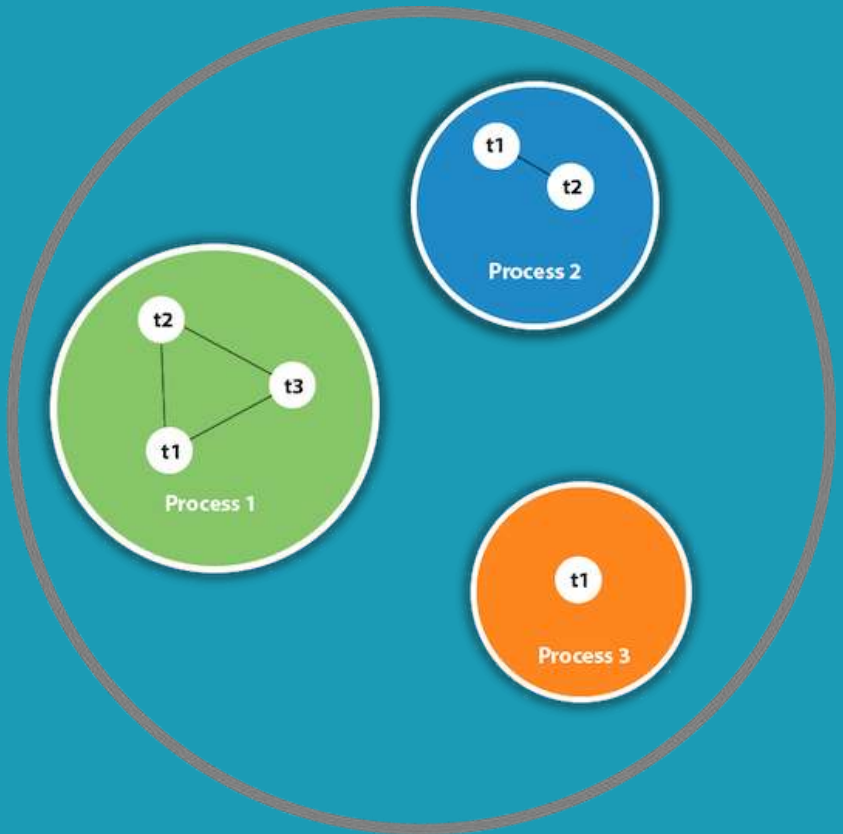
Ex: runs video, music and timer of VLC player simultaneously. It uses in Games, Animations etc.



Java: Multitasking, Multiprocessing & Multithreading

Characteristics of Thread

- Threads share the same address space.
- A thread is lightweight.
- Cost of communication between the thread is low.



Process based Multitasking

- Each process has an address in memory. In other words, each process allocates a separate memory area.
- A process is heavyweight.
- Cost of communication between the process is high.
- Switching from one process to another requires some time for saving and loading registers, memory maps, updating lists, etc.

'Thread based Multitasking

- Threads share the same address space.
- A thread is lightweight.
- Cost of communication between the thread is low.

Java: Multitasking, Multiprocessing & Multithreading

Multithreading

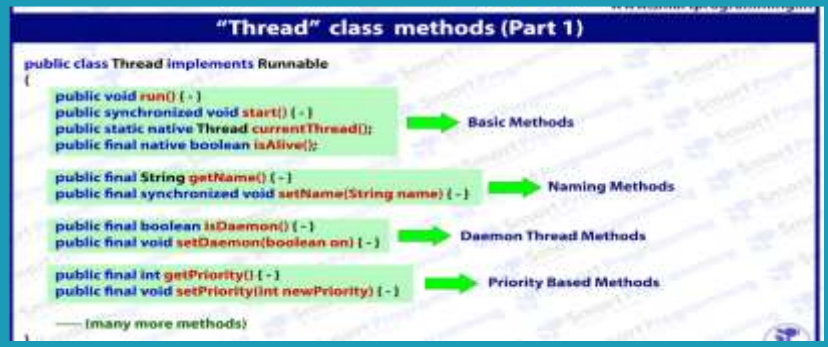
- It can be used in programming level
- Java provides predefined API for multithreading

Multithreading in Java

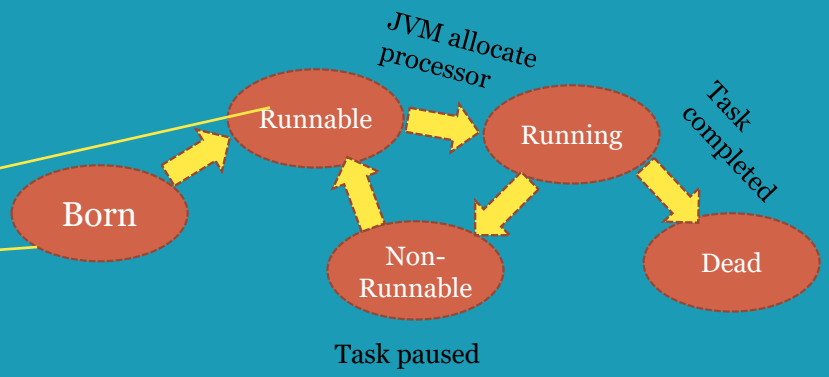
Threads can be created by using two mechanisms :
1.Extending the Thread class
2.Implementing the Runnable Interface

1. Thread creation by extending the Thread class
We create a class that extends the **java.lang.Thread** class. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread. Start() invokes the run() method on the Thread object. **Thread can not be start() more than once** as it is in the dead state, to run it again create another thread. It is 4-steps method.

```
class MyThread extends Thread{  //(Step-1)
     public void run(){  //(Step-2)
        System.out.println("Thread is Running");
     }
}
public class TestThread {
    public static void main(String[] args) {
        MyThread t=new MyThread(); //(Step-3)
        t.start(); //(Step-4)
    }
}
```



Life Cycle: Multithreading (5-stages)



Java: Multitasking, Multiprocessing & Multithreading

Thread creation by implementing the Runnable Interface
We create a new class which implements java.lang.Runnable interface and override run() method. Then we instantiate a Thread object and call start() method on this object.
Ex: In the following MyThread class implements Runnable. It is 5 steps method.

```
class MyThread implements Runnable{//(Step-1)
     public void run(){  //(Step-2)
        System.out.println("Thread is Running");
     }
}
public class TestThread {
    public static void main(String[] args) {
        MyThread t=new MyThread(); //(Step-3)
        Thread th=new Thread(t); //(Step-4)
        th.start(); //(Step-5)
    }
}
```

Single Task from Multiple Thread

```
class MyThread extends Thread{
     public void run(){
        System.out.println("Thread is Running");
     }
}
public class TestThread {
    public static void main(String[] args) {
        MyThread t1=new MyThread();
        MyThread t2=new MyThread();
        MyThread t3=new MyThread();
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Java: Multitasking, Multiprocessing & Multithreading

main Thread in Java

```
class VLCVideo extends Thread{
     public void run(){
        System.out.println(Thread.currentThread().getName());
        System.out.println("Video is Running");
     }
}
public class TestThread {
    public static void main(String[] args) {
        System.out.println("1="+Thread.currentThread().getName());
        Thread.currentThread().setName("Zaman");
        System.out.println("2="+Thread.currentThread().getName());
        VLCVideo t1=new VLCVideo();
        t1.start();
        System.out.println("3="+Thread.currentThread().getName());
    }
}
```

JVM automatically creates main thread and main creates another thread Thread-0.

Java: Multitasking, Multiprocessing & Multithreading

Multiple Task from Multiple Thread

```
class VLCVideo extends Thread{
     public void run(){
        for(int i=1;i<10;i++)
        System.out.println("Video is Running by "+Thread.currentThread().getName());
     }
}
class VLCMusic extends Thread{
    public void run(){
        for(int i=1;i<10;i++)
        System.out.println("Music is Running by"+Thread.currentThread().getName());
    }
}
class VLCTimer extends Thread{
    public void run(){
        for(int i=1;i<10;i++)
        System.out.println("Timer is Running by"+Thread.currentThread().getName());
    }
}
public class TestThread {
    public static void main(String[] args) {
        VLCVideo t1=new VLCVideo();
        VLCMusic t2=new VLCMusic();
        VLCTimer t3=new VLCTimer();
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Look at the outputs carefully, four threads are running simultaneously namely as miain, Thread-0, Thread-1 and Thread-2

THANK YOU