OOPJ Notes Day-12 Date: 10/05/2023

Multithreading in Java

- Creating Thread Using Thread Class

```
class Demo5 extends Thread
{
    public static int a=30;

    synchronized public void Print()
    {
        System.out.println("Print");
    }
    /*
    @Override
    public void run()
    {
        for(int i=0;i<50;i++)
        {
            if(currentThread().getName()=="Thread 2")
            {
            a++;
            System.out.println("I am Running Thread of Demo5:  "+a+" ThreadName: 
"+currentThread());
            }
            else
            {
                a--;
                System.out.println("I am Running Thread of Demo5:  "+a+" ThreadName: 
"+currentThread());
            }
        }
    }
    */
    @Override
    public void run()
    {
        for(int i=0;i<50;i++)
        {
            if(currentThread().getName()=="Thread 2")
            {
            System.out.println("I am Running Thread of Demo 5: ThreadName: 
"+currentThread());
            Print();
            }
            else
            {
                System.out.println("I am Running Thread of Demo5: ThreadName: 
"+currentThread());
                Print();
            }
        }
    }
}
public class ThreadClassDemo {

    public static void main(String[] args) {
        ThreadGroup tg=new ThreadGroup("Thread of Demo 5");
        Thread t1=new Demo5();
        t1.start();
        Thread t2=new Thread(tg,new Demo5(), "Thread 2");
        t2.start();




    }

}
```

- Types of Thread and their difference: User Thread versus Daemon Thread
- Thread termintation: Successfull or Unsuccessful completion of Thread Task
- Race condition and synchronized keyword
- Inter thread communication using wait,notify/notifyAll
    - Thread creation using Runnable versus Thread class
    - Blocking calls in Thread: interrupt() / interrupted() / isInterrupted()
    - Thread Priority
    - Joining Thread: use of join() method
    - Use of sleep() method
    - Exceptions related to Threads
    - Race Condition and Synchronized modifier: synchronized keyword: Concept of Mutual Exclusion
        - synchronozed instance variable
        - synchronozed instance method
        - synchronozed block
        - synchronozed static block
    - Deadloack situation in Multiple Threads
    - Inter Thread Communication: Use of: wait(), notify()/notifyAll() methods

```java
class Demo7 extends Thread
{
    synchronized public  void print()
    {
        for(int i=0;i<10;i++)
        {
            if(currentThread().getName()=="TH-1" && i==5)
            {
                try
                {
                System.out.println("TH-1 goint to sleep for 30000 ms");
                currentThread().wait(30000);
                }
                catch(Exception ex)
                {
                    System.out.println("Intrupted Exception occurred: "+ex.getMessage());
                }
                if(currentThread().getName()=="TH-2" && i==7)
                {

                    try {
                        notify();
                        System.out.println("I have sent notify and i am going for sleep");
                        currentThread().sleep(40000);
                    } catch (InterruptedException e) {
                        System.out.println(" "+e.getMessage());
                        //e.printStackTrace();
                    }

                }
            }
        System.out.println(i+" Print By:"+currentThread().getName());
        }
    }

    @Override
     public void run()
    {
        print();
    }
}
public class TGDemo2 {

    public static void main(String[] args) {

        Demo7 d1=new Demo7();

        ThreadGroup tg=new ThreadGroup("Demo T Thread");

        Thread t1=new Thread(tg, d1, "TH-1");
        Thread t2=new Thread(tg, d1, "TH-2");
        //Thread t3=new Thread(tg, d1, "TH-3");
        t1.start();
        t2.start();
        //t3.start();

}

}
```

try with resource

- External resources like File, Database con and N/W con to be in try block before its use.
- Will be discussed in Java I/O and JDBC, Java Socket Programming