Dt : 28/10/2023

**imp*

Creating and Executing thread:

step-1 : The user defined classes must be implemented from "java.lang.Runnable"

interface

Structure of Runnable:

public interface java.lang.Runnable

{

public abstract void run();

}

step-2 : The user defined implementation classes must construct body for run() method

step-3 : Create objects for User defined implementation classes

step-4 : Create object for "java.lang.Thread' class and while object creation pass user defined implementation class object ref as parameter.

step-5 : Execute run() method using start() method

Ex-program:

p1 : Register.java

```
package p1;
import java.util.*;
```

```
public class Register implements Runnable
{
     @Override
     public void run()
     {
     for(int i=1;i<=5;i++)
     {
          System.out.println("Reg..by Alex..."+new
Date());
          try {
               Thread.sleep(2000);//stop execution for
2000 milliSecs
          }catch(Exception e) {e.printStackTrace();}
     }
     }
}
```

p1 : Login.java

```
package p1;
import java.util.Date;
public class Login implements Runnable
{
     @Override
     public void run()
     {
     for(int i=1;i<=5;i++)
     {
          System.out.println("Log..by Ram..."+new
Date());
          try {
               Thread.sleep(2000);//stop execution for
2000 milliSecs
          }catch(Exception e) {e.printStackTrace();}
     }
     }
}
```

p2 : DemoThread1.java(MainClass)

```
package p2;
import p1.*;
public class DemoThread1
{
     public static void main(String[] args)
     {
          Register rg = new Register();
          Login lg = new Login();

          Thread t1 = new Thread(rg);
          Thread t2 = new Thread(lg);

          t1.start();
          t2.start();
     }
}
```

o/p:

Log..by Ram...Sat Oct 28 18:56:07 IST 2023

Reg..by Alex...Sat Oct 28 18:56:07 IST 2023

Log..by Ram...Sat Oct 28 18:56:09 IST 2023

Reg..by Alex...Sat Oct 28 18:56:09 IST 2023

Log..by Ram...Sat Oct 28 18:56:11 IST 2023

Reg..by Alex...Sat Oct 28 18:56:11 IST 2023

Log..by Ram...Sat Oct 28 18:56:13 IST 2023

Reg..by Alex...Sat Oct 28 18:56:13 IST 2023

Log..by Ram...Sat Oct 28 18:56:15 IST 2023

Reg..by Alex...Sat Oct 28 18:56:15 IST 2023

Diagram:

==========

****imp***

Creating threads using Anonymous InnerClasses:

=>In this model we can declare User defined implementation classes as Anonymous,

which means without name.

Ex-program:

p2 : DemoThread2.java(MainClass)

```
package p2;
import java.util.Date;
public class DemoThread2
{
     public static void main(String[] args)
     {
          //Register as Anonymous class
         Runnable rg = new Runnable()
         {
          @Override
             public void run()
             {
              for(int i=1;i<=5;i++)
              {
                   System.out.println("Reg..by
Alex..."+new Date());
                   try {
                        Thread.sleep(2000);//stop
execution for 2000 milliSecs
```

```
                    }catch(Exception e)
{e.printStackTrace();}
                }
               }
          };
          //Login as Anonymous class
          Runnable lg = new Runnable()
          {
           @Override
               public void run()
               {
                for(int i=1;i<=5;i++)
                {
                     System.out.println("Log..by
Ram..."+new Date());
                     try {
                          Thread.sleep(2000);//stop
execution for 2000 milliSecs
                     }catch(Exception e)
{e.printStackTrace();}
                }
               }
          };

          Thread t1 = new Thread(rg);
          Thread t2 = new Thread(lg);

          t1.start();
          t2.start();
     }
}
```

==========

*imp

Creating threads using LambdaExpressions:(Java8-model)

=>In this model we declare User defined implementation classes as LambdaExpressions

Ex-Program:

p2 : DemoThread3.java(MainClass)

```
package p2;
import java.util.Date;
public class DemoThread3
{
     public static void main(String[] args)
     {
          //Register as LambdaExpression
          Runnable rg = ()->
               {
                for(int i=1;i<=5;i++)
                {
                     System.out.println("Reg..by
Alex..."+new Date());
                     try {
                          Thread.sleep(2000);//stop
execution for 2000 milliSecs
                     }catch(Exception e)
{e.printStackTrace();}
                }
               };
          //Login as LambdaExpression
          Runnable lg = ()->
               {
                for(int i=1;i<=5;i++)
                {
                     System.out.println("Log..by
Ram..."+new Date());
                     try {
                          Thread.sleep(2000);//stop
execution for 2000 milliSecs
```

```
                    }catch(Exception e)
{e.printStackTrace();}
                }
               };

          Thread t1 = new Thread(rg);
          Thread t2 = new Thread(lg);

          t1.start();
          t2.start();
     }
}
```

==========

*imp

Creating threads using Method references:

Ex-program:

p1 : Register.java

```
package p1;
import java.util.*;
public class Register
{
     public static void reg()
     {
      for(int i=1;i<=5;i++)
      {
           System.out.println("Reg..by Alex..."+new
Date());
           try {
```

```
            Thread.sleep(2000);//stop execution for
2000 milliSecs
          }catch(Exception e) {e.printStackTrace();}
     }
     }
}
```

p1 : Login.java

```
package p1;
import java.util.Date;
public class Login
{
    public static void log()
     {
     for(int i=1;i<=5;i++)
     {
          System.out.println("Log..by Ram..."+new
Date());
          try {
               Thread.sleep(2000);//stop execution for
2000 milliSecs
          }catch(Exception e) {e.printStackTrace();}
     }
     }
}
```

p2 : DemoMethods4.java(MainClass)

```
package p2;
import p1.*;
public class DemoThread4
{
     public static void main(String[] args)
     {
          Runnable rg = Register :: reg;
          Runnable lg = Login :: log;

          Thread t1 = new Thread(rg);
```

```
        Thread t2 = new Thread(lg);

        t1.start();
        t2.start();
    }
}
```

==========