* Explain Thread Create Method.

=> There are two method to create a thread.

1) By extending thread class

2) By implementing Runnable interface.

1 By extending thread class:

Thread class provides a different method to perform different operations.

Syntax:

```
class class extends Thread
      name
```

Ex.

```
class F extends Thread
{
  public void run()
  {
    System.out.println(" Thread ");
  }
```

```
public static void main
    (String. args[])
{
    F obj = new F();

        obj.start();
}

}
```

2 By implementing Runnable interface.

=> The runnable interface can implement the thread.

Syntax:

```
class class implements
      name
                Runnable
```

Ex.

```
class F implements Runnable
{
  public void run()
{
  System.out.println("Thread");
}
```

```
public static void main(String
        args[])
{
    F obj = new F();
  Thread t1 = new Thread(obj);

        t1.start();
}

}
```

* Explain different methods of Thread.

=> This are the different methods of Thread.

1 void run() : Used to perform action of a thread.

2 void start() : Starts the excution of a thread.

3 int getPriority() : Returns the Priority of a thread.

4 int setPriority() : Changes the Priority of a thread.

5 int getId() : Returns id of the thread.

6 String getName() : Returns the name of the thread.

7 String setName() : Changes the name of the thread.

8 Thread.State.getState() : Returns the state of the thread.

Ex.

```
import java.lang.*;

class java extends Thread
{
  public void run()
  {
    System.out.println("Thread");
  }
  public static void main(String.
      args[])
  {
    System.out.println("Current
    thread name" + Thread.
    currentThread().getName());
```

```
System.out.println("Current thread Priority " + Thread.currentThread.getPriority());

Thread t1 = new Thread();

System.out.println("Thread Priority " + t1.getPriority());

t1.setPriority(2);

System.out.println("Thread Priority " + t1.getPriority());

    }
}
```

* Explain Synchronization of Thread.

=> Synchronization of thread is used to control multithreading.

~~Using~~ Using Synchronization, we can access thread one by one and in serial manner.

There are three method to Perform thread Synchronization.

1) By Using Synchronization Method

2) By Using Synchronization Block

3) By Using Static synchronization

Ex.

```
class Java
{
 synchronized void print(int n)
 {
   for(int i=1; i<=5; i++)
   {
     System.out.println(n*i);
   try
   {
     Thread.sleep(400);
   }
   catch(Exception e)
   {
     System.out.println(e);
   }
   }
}
```

Page No.
Date : / /

```
class Thread1 extends Thread
{
    Java J;
    Thread1(Java J);
    this.J = J;

    public void run()
    {
        J.print(5);
    }
}
class Thread2 extends Thread
{
    Java J;
    Thread2(Java J);
    this.J = J;

    public void run()
    {
        J.print(10);
    }
}

class Java1
{
  public static void main(String args[])
  {
      Java obj = new Java();
      Thread1 t1 = new Thread1(obj);
      Thread2 t2 = new Thread2(obj);
```

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