Unit-4

Multi Threading

Processed based Multi tasking

Doing more than one task simultaneously(Multiple applications(programs) running at the same time)

Eg:

Listening to songs while typing java program in notepad

Thread based Multi tasking (Multi Threading)

A single program doing more than one task simultaneously.(Single application doing more tasks simultaneously)

Eg:

- Downloading a song while listening to it(youtube)
- All students requesting the server for result at a time
- playing the background music at the same time as playing the game is an example of multithreading.
- Joint Account holder having multiple ATM cards for same account and trying to perform operation same time
- Online bus/anything ticket booking : here many users trying to book available ticket at same time (ex : tatkal booking) , here application needs to handle different threads (diff users request to server)
- You are typing a paragraph on MS word. But in background one more thread running and checking your spelling mistakes.

---

main() is a thread

```
class inter
{
        public static void main(String cash[])
```

```
        {
                System.out.println("hello...");
                System.out.println(Thread.currentThread());
                System.out.println("bye");
        }
}
```

How to create a thread

By extending 'Thread' class
By implementing 'Runnable' interface

Example 1

```
class mul extends Thread
{
        public void run()
        {
                Thread name=Thread.currentThread();
                System.out.println(name.getName()+": entered");
                for(int i=1;i<=100;i++)
                {
                        System.out.println(i);
                }
                System.out.println(name.getName()+": left");
        }
}
class data
{
        public static void main(String cash[])
        {
                mul m=new mul();
                Thread t1=new Thread(m);
                Thread t2=new Thread(m);
                t1.setName("kiwi");
                t2.setName("yuvi");
                t1.start();
                t2.start();
                System.out.println("bye");
                System.out.println(Thread.currentThread().getName());
        }
}
```

Example 2

```
class mul extends Thread
{
	int n;
	mul(int val)
	{
		n=val;
	}
	public void run()
	{
		Thread name=Thread.currentThread();
		System.out.println(name.getName()+": entered");
		for(int i=1;i<=10;i++)
		{
			System.out.println(n+"*"+i+"="+n*i);
		}
		System.out.println(name.getName()+": left");
	}
}
class data
{
	public static void main(String cash[])
	{
		mul m1=new mul(10);
		mul m2=new mul(12);
		Thread t1=new Thread(m1);
		Thread t2=new Thread(m2);
		t1.setName("kiwi");
		t2.setName("yuvi");
		t1.start();
		t2.start();
		System.out.println("bye");
		System.out.println(Thread.currentThread().getName());
	}
}
```

Example 3

```
class num extends Thread
{

	public void run()
	{

		Thread name=Thread.currentThread();
		System.out.println(name.getName()+": thread entered");
		for(int i=1;i<=5;i++)
		{
			System.out.println(i+" by thread "+ name.getName());
			try
			{

				Thread.sleep(1000);

			}
			catch(Exception e)
			{

			}

		}
		System.out.println(name.getName()+": thread left");
	}
}
class app
{
	public static void main(String cash[])throws Exception
	{
		System.out.println(Thread.currentThread().getName()+": thread started");

		num m=new num();
		Thread t1=new Thread(m);
		Thread t2=new Thread(m);
		t1.setName("kiwi");
		t2.setName("yuvi");
		t1.start();
		t2.start();
		System.out.println("bye");
		System.out.println(Thread.currentThread().getName()+": thread ended");
	}
}
```

Synchronization

Using synchronized block

```
class multitasks extends Thread
{
	public void run()
	{
		synchronized(this)
		{
			System.out.print("[");
			try
			{
				Thread.sleep(1000);
			}
			catch(InterruptedException i)
			{

				System.out.print("disturbed.....");
			}
			System.out.print("svec"+"]");
			System.out.println();
		}
	}


}
class threaddemo
{
	public static void main(String many[])
	{
		multitasks m=new multitasks();
		Thread t1=new Thread(m);
		Thread t2=new Thread(m);
		Thread t3=new Thread(m);
		t1.start();
		t2.start();
```

```
                t3.start();
        }
}
```

Using synchronized method

EXAMPLE 1

```
class multitasks extends Thread
{
        public void run()
        {
                display();
        }
        synchronized void display()
        {
                        System.out.print("[");
                        try
                        {
                                Thread.sleep(1000);
                        }
                        catch(InterruptedException i)
                        {

                                System.out.print("disturbed.....");
                        }
                        System.out.print("svec"+"]");
                        System.out.println();
        }


}
class threaddemo
{
        public static void main(String many[])
        {
```

```
                multitasks m=new multitasks();
                Thread t1=new Thread(m);
                Thread t2=new Thread(m);
                Thread t3=new Thread(m);
                t1.start();
                t2.start();
                t3.start();
        }
}
```

Using synchronized block

```
class multitasks extends Thread
{
        static int bal=100;
        int cbal;
        public void run()
        {

                synchronized(this)
                {
                try
                {



                        for(int i=1;i<=5;i++)
                        {
                                cbal=bal+i;
                                Thread.sleep(100);
                                bal=cbal;
                                System.out.println(bal);
                        }
                }
                catch(Exception e)
                {

                }
                }

        }
```

```
}
class sync
{
	public static void main(String many[])throws Exception
	{

		multitasks m=new multitasks();
		Thread t1=new Thread(m);
		Thread t2=new Thread(m);
		System.out.println("before creditbal :"+multitasks.bal);
		t1.start();
		t2.start();
		t1.setName("credit 1");
		t2.setName("credit 2");
		t1.join();
		t2.join();
		System.out.println("after credit bal :"+multitasks.bal);

	}
}
```

Inter thread communication

Multithreading feature in Java allows concurrent execution of two or more parts of a program. Each part is a Thread.

EXAMPLE 1

```
class jointacc extends Thread
{

	static int bal=10000;
	public void run()
	{
		if(this.currentThread().getName()=="with")
		{
			withdraw(15000);
		}
		else
		{
			deposit(10000);
		}
	}
	synchronized void withdraw(int amount)
	{
		System.out.println("amount requested for withdrawn :"+amount);

		if(bal<amount)
		{
			System.out.println("current balance :"+bal);
			System.out.println("Less balance; waiting for deposit...");
			try
			{
				wait();
			}
			catch(Exception e)
			{
```

```
                        }
                }
                bal=bal-amount;
                System.out.println(amount+" rupees withdraw completed...");
                System.out.println("current balance :"+bal);
        }
        synchronized void deposit(int amount)
        {
                System.out.println("going to deposit...");
                bal=bal+amount;
                System.out.println(amount+" rupees deposit completed...");
                System.out.println("current balance :"+bal);
                notify();
        }
}

class sbi
{
        public static void main(String args[])
        {
                jointacc svec=new jointacc();
                Thread t1=new Thread(svec);
                Thread t2=new Thread(svec);
                t1.setName("with");
                t2.setName("depo");
                t1.start();
                t2.start();
        }
}
```

EXAMPLE 2

```
class ticktock extends Thread
{
        public void run()
        {
                if(this.currentThread().getName()=="tick")
                {
                        tick();
                }
                else
                {
                        tock();
                }
        }
        synchronized void tick()
        {
```

```
                for(int i=1;i<=5;i++)
                {
                        System.out.print("tick ");
                        notify();
                        try
                        {
                                wait();
                        }
                        catch(Exception e)
                        {
                        }
                        System.out.println("bye");

                }
        }
        synchronized void tock()
        {
                for(int i=1;i<=5;i++)
                {
                        System.out.println("tock");
                        notify();
                        try
                        {
                                wait();
                        }
                        catch(Exception e)
                        {
                        }

                }
        }
}

class sbi
{
        public static void main(String args[])
        {
                ticktock svec=new ticktock();
                Thread t1=new Thread(svec);
                Thread t2=new Thread(svec);
                t1.setName("tick");
                t2.setName("tock");
                t1.start();
                t2.start();

        }
}
```

OUTPUT

```
tick tock
bye
tick tock
bye
tick tock
bye
tick tock
bye
tick tock
bye
```

Note

One most important difference between wait() and join() that is wait() must be called from synchronized context i.e. synchronized block or method otherwise it will throw IllegalMonitorStateException but On the other hand, we can call join() method with and without synchronized context in Java.

Thread priorities

```
class num extends Thread
{

	public void run()
	{

		Thread name=Thread.currentThread();
		System.out.println(name.getName()+": thread entered");
		for(int i=1;i<=5;i++)
		{
			System.out.println(i+" by thread "+ name.getName());
			try
			{

				Thread.sleep(1000);

			}
			catch(Exception e)
			{

			}

		}
```

```
                System.out.println(name.getName()+": thread left");
        }
}
class app
{
        public static void main(String cash[])throws Exception
        {
                Thread.currentThread().setName("svec");
                System.out.println(Thread.currentThread().getName()+": thread started");
                num m=new num();
                Thread t1=new Thread(m);
                Thread t2=new Thread(m);
                Thread t3=new Thread(m);
                t1.setPriority(1);
                t2.setPriority(5);
                t3.setPriority(10);
                t1.setName("kiwi");
                t2.setName("yuvi");
                t3.setName("buvi");
                t1.start();
                t2.start();
                t3.start();
                System.out.println("bye");
                System.out.println(Thread.currentThread().getName()+": thread ended");
        }
}
```