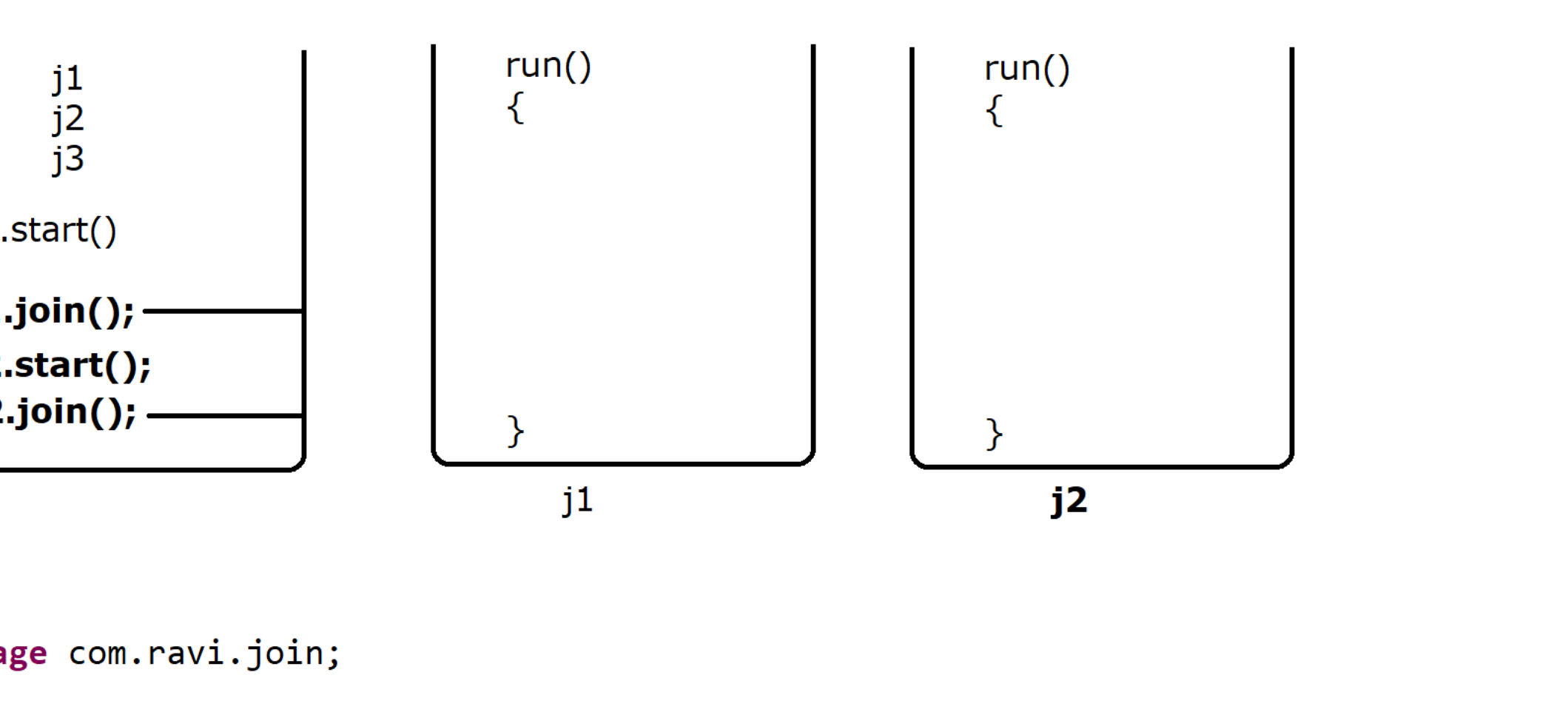
public final void join() throws InterruptedException :
----

* It is a predefined final and non static method of Thread class.

* The main purpose of this method to put the **current thread into temporarly waiting state till the completion of another thread on which we have invoked join() method.**

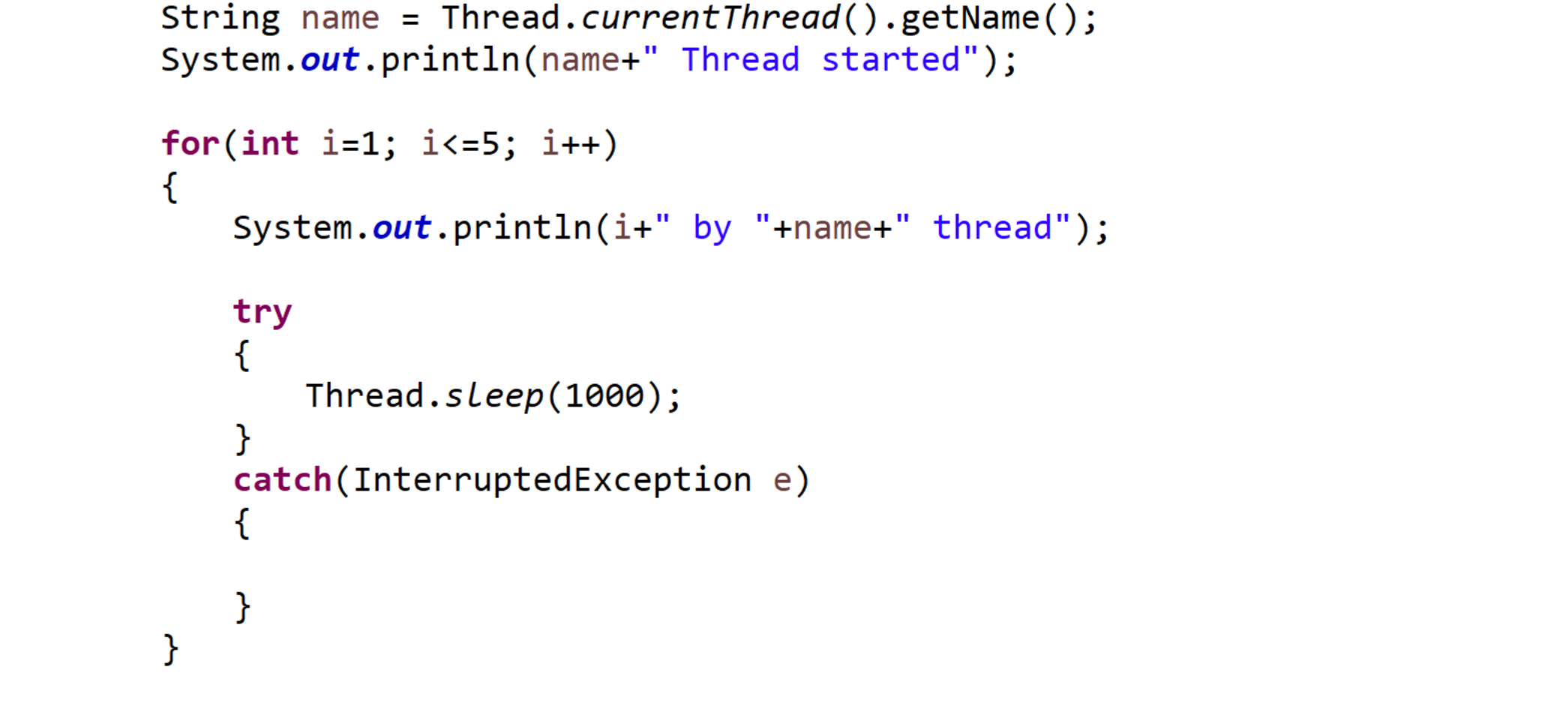
* It also throws checked exception i.e. InterruptedException so, provide try-catch OR declare the method as throws.



In this diagram, t1 thread (Withdraw thread) is waiting for t2 thread (deposit thread) to complete, Once the t2 thread (deposit thread) will go to DEAD state then only t1 thread can complete its remaining task.

ALWAYS THE THREAD WHICH IS CALLING JOIN() METHOD WILL BE COMPLETED FIRST

//Program :



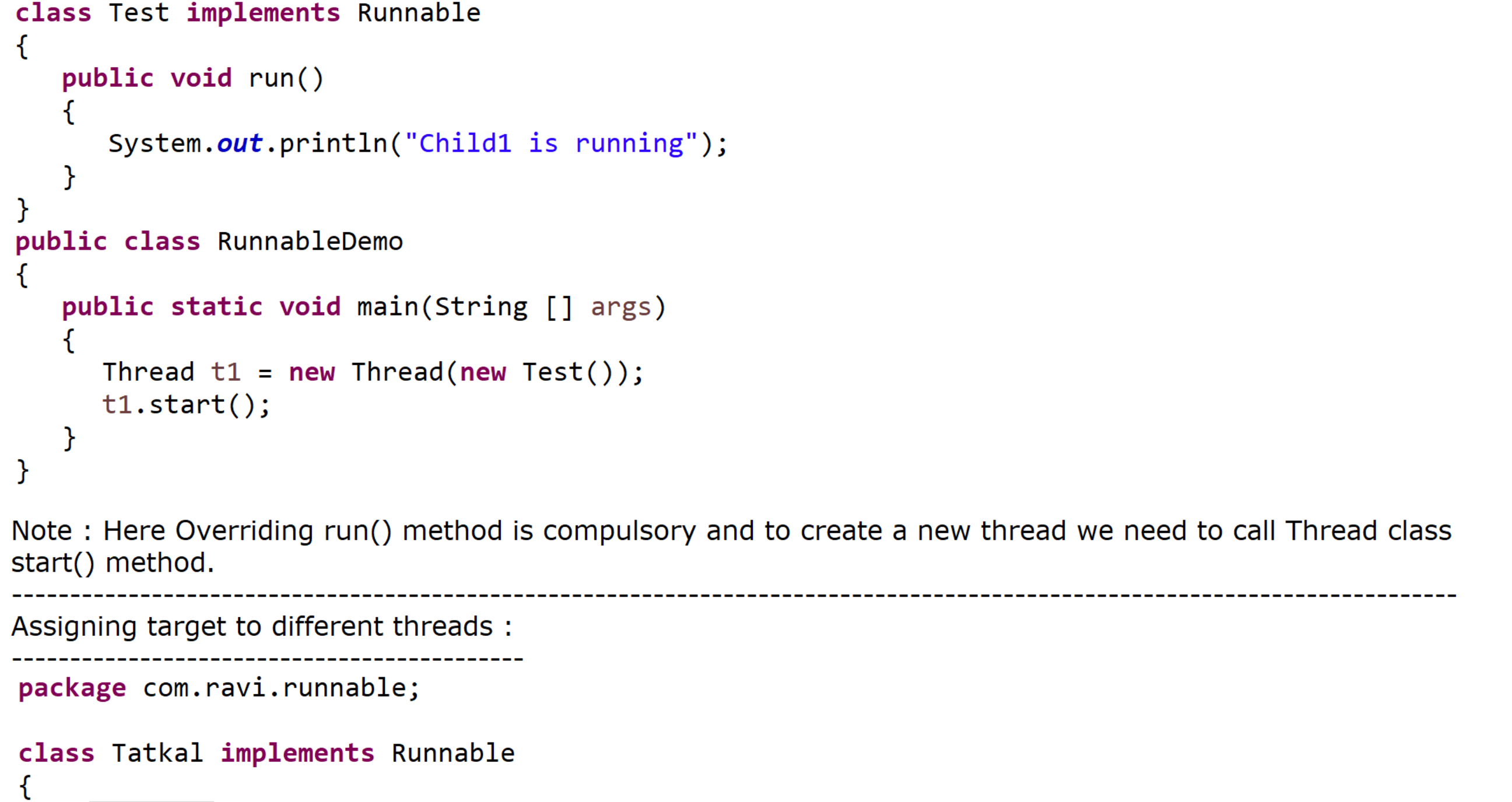

```java
package com.ravi.join;

class Join extends Thread
{
    @Override
    public void run()
    {
        String name = Thread.currentThread().getName();
        System.out.println(name+" Thread started");

        for(int i=1; i<=5; i++)
        {
            System.out.println(i+" by "+name+" thread");

            try
            {
                Thread.sleep(1000);
            }
            catch(InterruptedException e)
            {

            }
        }

        System.out.println(name+" thread completed!!!");

    }
}

public class JoinDemo1
{
    public static void main(String[] args)
    {
        System.out.println("Main Thread Started!!!");

        Join j1 = new Join();
        Join j2 = new Join();
        Join j3 = new Join();

        j1.setName("J1");
        j2.setName("J2");
        j3.setName("J3");


        j1.start();

        try
        {
            j1.join(); //main thread is in waiting for j1 thread to complete
        }
        catch(InterruptedException e)
        {
            System.err.println(e);
        }

        j2.start();
        j3.start();

        System.out.println("Main Thread Ended!!!");

    }

}
```

```java
package com.ravi.join;

class Alpha extends Thread
{
    @Override
    public void run()
    {
        Thread t = Thread.currentThread();
        String name = t.getName();   //Alpha_Thread is current thread

        Beta b1 = new Beta();
        b1.setName("Beta_Thread");
         b1.start();
         try
         {
            b1.join(); //Alpha thread is waiting 4 Beta Thread to complete

            System.out.println("Alpha thread re-started");
        }
         catch (InterruptedException e)
         {
            e.printStackTrace();
        }

        for(int i=1; i<=10; i++)
        {
            System.out.println(i+" by "+name);
        }

    }
}

public class JoinDemo2
{
    public static void main(String[] args)
    {
        Alpha a1 = new Alpha();
        a1.setName("Alpha_Thread");
        a1.start();
    }
}

class Beta extends Thread
{
    @Override
    public void run()
    {
        Thread t = Thread.currentThread();
        String name = t.getName();   //Beta_Thread

        for(int i=1; i<=20; i++)
        {
            System.out.println(i+" by "+name);
            try
            {
                Thread.sleep(500);
            }
            catch(InterruptedException e) {

            }
        }
        System.out.println("Beta Thread Ended");
    }
}
```

```java
package com.ravi.join;

public class JoinDemo3
{
    public static void main(String[] args) throws InterruptedException
    {
        System.out.println("Main Thread started");

        Thread t = Thread.currentThread();

        for(int i=1; i<=5; i++)
        {
            System.out.println(i+" by "+t.getName());
            Thread.sleep(1000);
        }

        t.join(); //main thread is waiting for main thread to complete (Deadlock)

        System.out.println("Main thread ended...");

    }
}
```

```java
package com.ravi.join;
//Online banking amount transfer after OTP Verification

class OTPVerification extends Thread
{
    public void run()
    {
        System.out.println("Verifying OTP...");
        try
        {
            Thread.sleep(2000);
        }
        catch (InterruptedException e)
        {
            System.err.println("Thread is Interrupted");
        }
        System.out.println("OTP Verified..");
    }
}

class BalanceCheck extends Thread
{
    public void run()
    {
        System.out.println("Checking account balance...");
        try
        {
            Thread.sleep(1500);
        }
        catch (InterruptedException e)
        {
            System.err.println("Thread is Interrupted");
        }
        System.out.println("Sufficient Balance Available...");
    }
}
public class JoinDemo4
{
    public static void main(String[] args) throws InterruptedException
    {
        OTPVerification otpThread = new OTPVerification();
        BalanceCheck balanceThread = new BalanceCheck();

        otpThread.start();
        balanceThread.start();

        otpThread.join();
        balanceThread.join();

        //If both the threads completed successfully then only main thread will proceed.

        System.out.println("Initiating Money Transfer...");
        System.out.println("Transfer Successful...");
    }
}
```

Assigning the target for the thread by using Runnable interface : (2nd Approach)



```java
package com.ravi.runnable;

class Test implements Runnable
{
   public void run()
   {
      System.out.println("Child1 is running");
   }
}
public class RunnableDemo
{
   public static void main(String [] args)
   {
      Thread t1 = new Thread(new Test());
      t1.start();
   }
}
```

Note : Here Overriding run() method is compulsory and to create a new thread we need to call Thread class start() method.

Assigning target to different threads :

```java
package com.ravi.runnable;

class Tatkal implements Runnable
{
    @Override
    public void run()
    {
       String name = Thread.currentThread().getName();
       System.out.println(name+" has booked the ticket under Tatkal Scheme");
    }
}

class PremimumTatkal implements Runnable
{
    @Override
    public void run()
    {
        String name = Thread.currentThread().getName();
        System.out.println(name+" has booked the ticket under Premimum Tatkal Scheme");
    }
}

public class TicketBooking
{
    public static void main(String[] args)
    {
        Thread scott = new Thread(new Tatkal(),"Mr. Scott");
        scott.start();

        Thread alen = new Thread(new PremimumTatkal(), "Mr. Alen");
        alen.start();
    }
}
```