Designing and Using the Thread Class

In Java, you can create and use threads by extending the `Thread` class or implementing the Runnable interface. Here's a simple example that demonstrates both approaches.

Example:

1. Using the Thread Class

In this example, we'll create a custom thread by extending the `Thread` class. The thread will print numbers from 1 to 5.

```
// Custom Thread class
class MyThread extends Thread
{
   @Override
   public void run()
   {
      for (int i = 1; i <= 5; i++)
     {
         System.out.println("Thread: " + i);
         try
         {
                  Thread.sleep(500); // Sleep for 500 milliseconds
         }
         catch (InterruptedException e)
         {
                  System.out.println("Thread interrupted: " + e.getMessage());
         }
      }
   }
}
```

```
// Main class
public class ThreadExample
{
   public static void main(String[] args)
   {
      MyThread thread = new MyThread(); // Create a new thread
      thread.start(); // Start the thread

      // Main thread printing numbers
      for (int i = 1; i <= 5; i++)
      {
         System.out.println("Main: " + i);
         try
         {
                  Thread.sleep(300); // Sleep for 300 milliseconds
         }
         catch (InterruptedException e)
         {
                  System.out.println("Main thread interrupted: " + e.getMessage());
         }
      }
   }
}
```

Explanation:

1. Custom Thread Class (MyThread):

- Extends `Thread` and overrides the *run*() method to define what the thread will do.

- It prints numbers from 1 to 5 and sleeps for 500 milliseconds between prints.

2. Main Class (*ThreadExample*):

- Creates an instance of *MyThread* and starts it with *thread.start*().

- The main thread also prints numbers from 1 to 5, but sleeps for 300 milliseconds between prints.

2. Using the Runnable Interface

Let's see how to implement the `Runnable` interface to achieve the same functionality.

```
// Custom Runnable class
class MyRunnable implements Runnable
{
   @Override
   public void run()
   {
      for (int i = 1; i <= 5; i++)
      {
         System.out.println("Runnable: " + i);
         try
         {
                  Thread.sleep(500); // Sleep for 500 milliseconds
         }
         catch (InterruptedException e)
         {
                  System.out.println("Runnable interrupted: " + e.getMessage());
         }
      }
   }
}

// Main class
public class RunnableExample
{
   public static void main(String[] args)
   {
      MyRunnable myRunnable = new MyRunnable(); // Create a new Runnable
      Thread thread = new Thread(myRunnable); // Create a thread using Runnable
      thread.start(); // Start the thread
```

```
        // Main thread printing numbers
        for (int i = 1; i <= 5; i++)
        {
                System.out.println("Main: " + i);
                try
                {
                        Thread.sleep(300); // Sleep for 300 milliseconds
                }
                catch (InterruptedException e)
                {
                        System.out.println("Main thread interrupted: " + e.getMessage());
                }
        }
    }
}
```

Explanation:

1. Custom Runnable Class (*MyRunnable*):

- Implements the *Runnable* interface and provides the implementation for the *run*() method.

- Similar behavior as before, printing numbers from 1 to 5 with a 500 milliseconds sleep.

2. Main Class (*RunnableExample*):

- Creates an instance of *MyRunnable* and wraps it in a *Thread* object.

- Starts the thread, and the main thread also prints numbers.

Running the Programs

When you run either of the above programs, you will see the output from both the main thread and the custom thread (or runnable) interleaved, as they run concurrently. The exact order may vary due to the thread scheduling by the Java Virtual Machine (JVM).

Feel free to modify the sleep durations or the range of numbers to see different behaviors!