A **thread** is a lightweight, independent path of execution inside a program — letting it handle multiple tasks at the same time.

A **thread** is the smallest unit of a **process** that can run independently.

- A **process** is a running program (e.g., your browser, a game, or a text editor).
- A **thread** is like a mini-task inside that program.

📌 Think of a thread as a **single line of work** the program is doing.

🔄 Why Use Threads?

Because they allow the program to **do more than one thing at a time**.

Simple Example for Beginners:

Imagine you're making tea :

- **Boil water** – one task (Thread 1)
- **Clean cups** – another task (Thread 2)

If you do them **one by one**, it takes longer.
If you do them **at the same time**, it's faster — that's **multithreading**!

In Java, each of those tasks would run in a separate thread.

The **main thread** is created automatically by the **Java Virtual Machine (JVM)** when your program starts, and it runs the main() method.

We can not see the java main thread class, we just Extend it or use a Runnable interface to use it.

🔧 How to Create Threads in Java

✅ 1. Extending the Thread class

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running...");
    }
}
public class Main {
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        t1.start();
    }
}
```

- run() → defines the task
- start() → actually starts a new thread and calls run()

✅ 2. Implementing Runnable interface

```
class MyRunnable  implements Runnable {
   public void run() {
      System.out.println("Task is running in thread: " + Thread.currentThread().getName());
   }
}

public class RunnableExample {
   public static void main(String[] args) {
      MyRunnable task = new MyRunnable();
      Thread t = new Thread(task);
      t.start();

      System.out.println("Main thread: " + Thread.currentThread().getName());
   }
}
```

✔️ Recommended way — because you can extend another class too

Important Thread Methods

- **start()**

Starts a new thread — calls the run() method in a new call stack

- **run()**

Contains the code that the thread will execute

- **sleep(ms)**

Pauses the thread for a certain time (in milliseconds)

```
Thread.sleep(1000); // pauses for 1 second
```

- **join()**

Makes one thread wait for another to finish

```
t1.join();
```

- **isAlive()**

Checks if a thread is still running

```
if (t1.isAlive()) {
    System.out.println("Still running...");
}
```

- **interrupt()**

Used to signal a thread to stop or break out of sleep/wait

```
t1.interrupt();  // sends an interrupt signal
```

In the thread, you check:

```
if (Thread.interrupted()) {
    System.out.println("Thread interrupted!");
}
```

Multiple Threading

```
// Thread that prints numbers from 1 to 5
class NumberThread extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Number: " + i);
        }
    }
}

// Thread that prints letters from 'A' to 'E'
class LetterThread extends Thread {
    public void run() {
        for (char ch = 'A'; ch <= 'E'; ch++) {
            System.out.println("Letter: " + ch);
        }
    }
}

public class ThreadExample {
  public static void main(String[] args) {
        // Create instances of both threads
        NumberThread n = new NumberThread();
        LetterThread l = new LetterThread();

        // Start both threads
        n.start();  // Starts the NumberThread
        l.start();  // Starts the LetterThread
    }
}
```

Thread Independence: NumberThread and LetterThread are running at the same time, and their output might **interleave** randomly.

Thread Scheduling: The JVM decides when each thread gets CPU time, so the order in which the output is printed may change each time you run the program.

Output can be -	**Or it can be -**
Number: 1	Letter: A
Letter: A	Letter: B
Number: 2	Letter: C
Letter: B	Letter: D
Number: 3	Letter: E
Letter: C	Number: 1
Number: 4	Number: 2
Letter: D	Number: 3
Number: 5	Number: 4
Letter: E	Number: 5

Lecture

by

Maliha Bushra Hoque (MBH)