🧵 Ways to Create a Thread in Java

1. By Extending `Thread` Class

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread using Thread class");
    }
}
new MyThread().start();
```

2. By Implementing `Runnable` Interface

```
class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Thread using Runnable");
    }
}
Thread t = new Thread(new MyRunnable());
t.start();
```

3. By Implementing `Callable` Interface (Java 5+)

```
Callable<String> task = () -> "Callable executed";
ExecutorService executor = Executors.newSingleThreadExecutor();
Future<String> future = executor.submit(task);
System.out.println(future.get());
executor.shutdown();
```

Differences: `Thread` vs `Runnable` vs `Callable`

Feature	`Thread`	`Runnable`	`Callable<V>`
Inheritance	Extends `Thread`	Implements interface	Implements interface
Return Value	No	No	Yes (`V`)
Throws Exception	No	No	Yes (can throw checked exceptions)

Feature	`Thread`	`Runnable`	`Callable<V>`
Execution	`start()`	Pass to `Thread` → `start()`	Submit to `ExecutorService`
Method to Implement	`void run()`	`void run()`	`V call() throws Exception`
Thread Reuse	Not reusable	Reusable in multiple threads	Reusable via `ExecutorService`

Key Methods

`Thread`:

- `start()`
- `run()`
- `sleep()`
- `join()`

`Runnable`:

- `run()`

`Callable<V>`:

- `call()`

📷Functional Interfaces

- `Runnable` is a **functional interface** (has one abstract method `run()`).
- `Callable` is also a **functional interface** (has one abstract method `call()`).

This allows both to be used with lambda expressions in Java 8+.

Can You Use `Callable` Independently?

No, `Callable` cannot be executed directly or with `Thread`. It must be used with `ExecutorService` (or similar concurrency utility).

What is `ExecutorService`?

- An interface in `java.util.concurrent` to manage a **pool of threads**.
- Handles:
- Thread lifecycle
- Task submission (`Runnable` or `Callable`)
- Graceful shutdown
- Result tracking via `Future`

Code Snippet: Using `ExecutorService`

```
import java.util.concurrent.*;

public class ExecutorExample {
    public static void main(String[] args) throws Exception {
        ExecutorService executor = Executors.newFixedThreadPool(2);

        Callable<String> task = () -> {
            Thread.sleep(1000);
            return "Callable Task Completed";
        };

        Future<String> future = executor.submit(task);
        System.out.println("Result: " + future.get());

        executor.shutdown();
    }
}
```
