**Day 10 docs**

**Task -1:**

**What is a process?**

A process is a running program.

like :

* When you click on an app (like a web browser or a music player), your computer starts a process for it.
* That process runs the app and uses your computer’s memory and CPU to do its job.

### Example:

1. The calculator app on your phone is a program.

2. When you open it, your phone starts a process for the calculator.

3. That process runs the calculator and lets you do math.

**Task - 2:**

**What is a Thread?**

A thread is a small part of a process that does a specific task.

Think of it like:

1. A process is like a team working on a project.

2. Each thread is a team member doing one job.

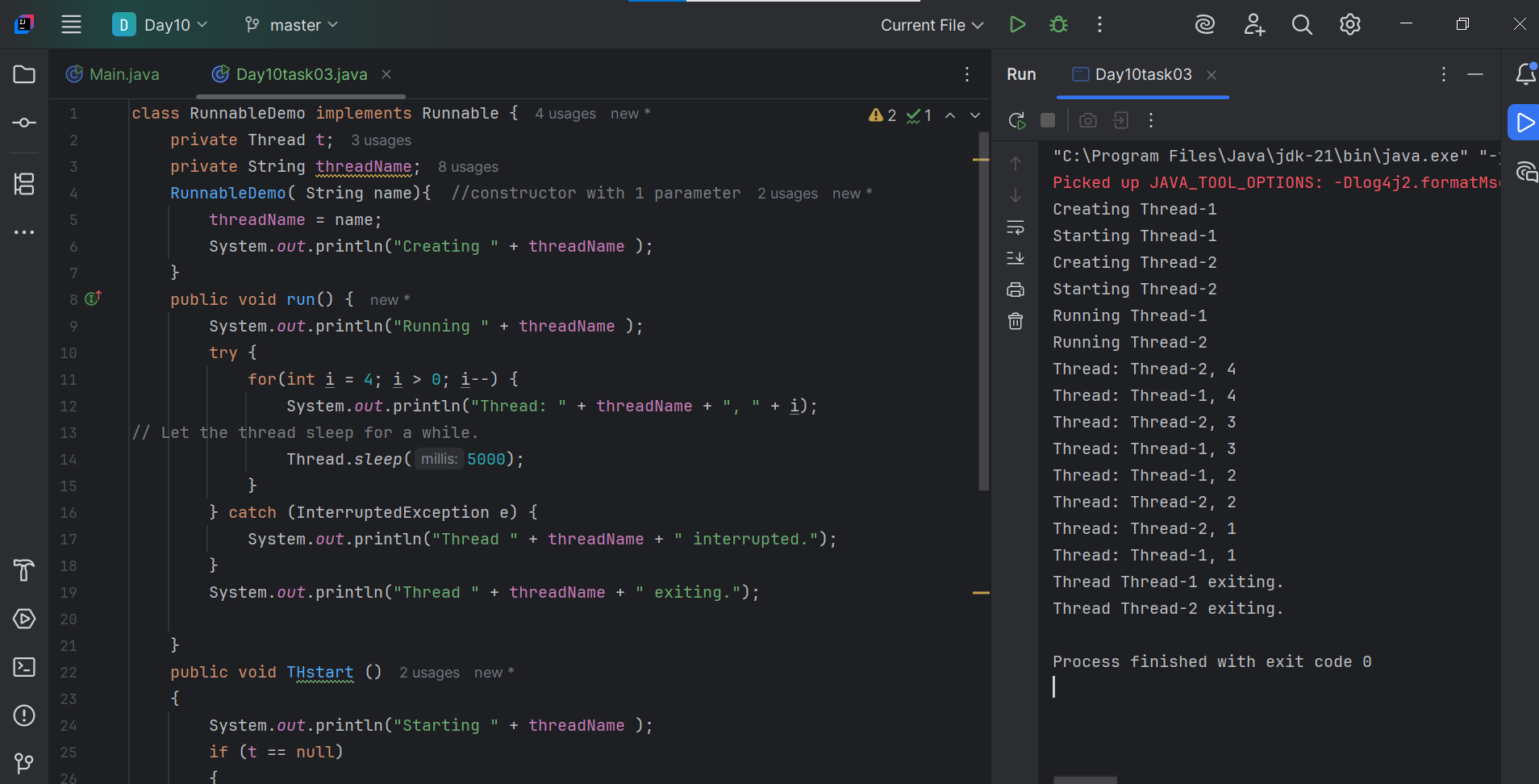
3. All threads share the same space and resources (like memory), but they work on different tasks at the same time.

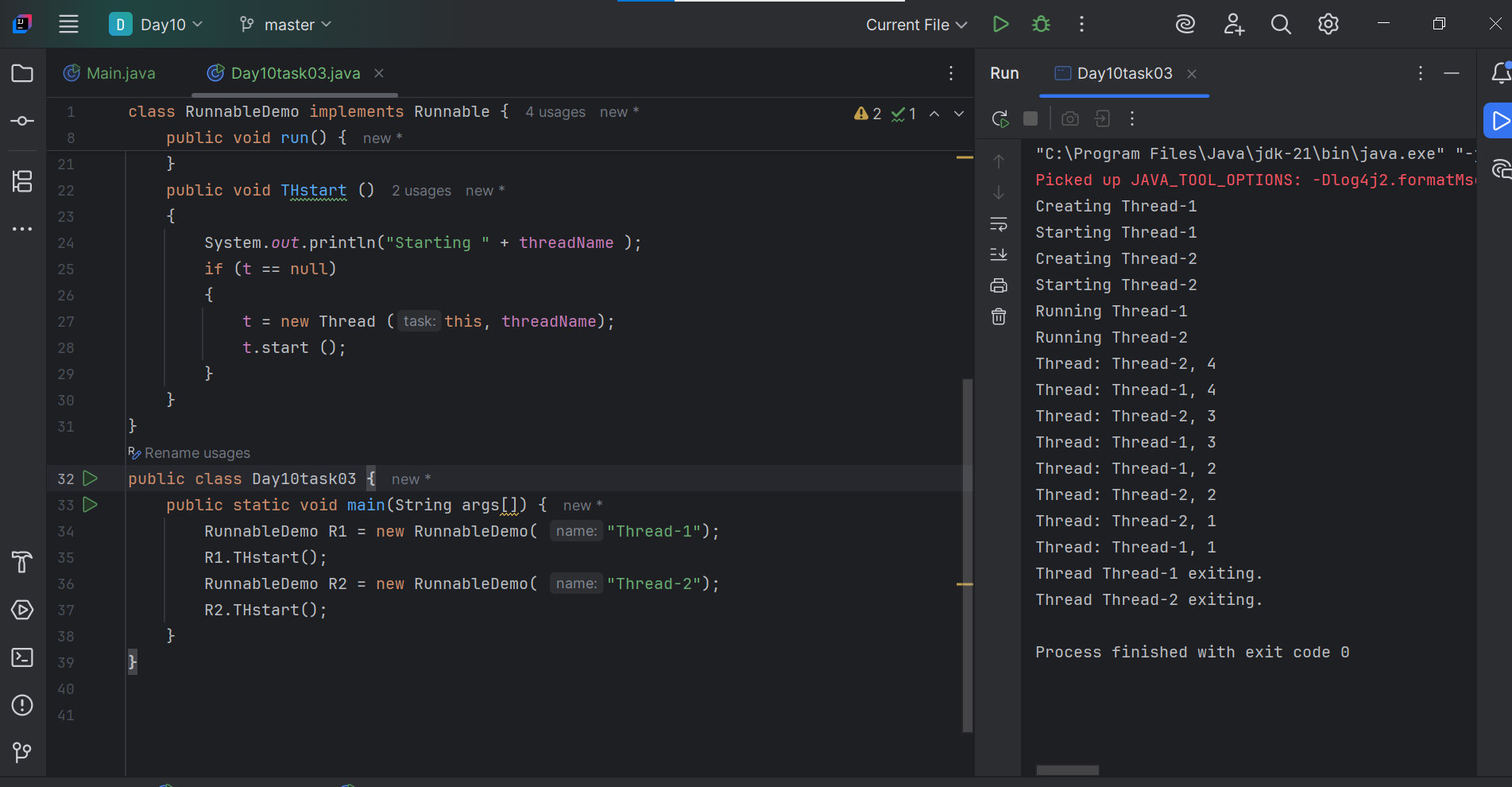
### Example:

Imagine your web browser is a process. Inside it

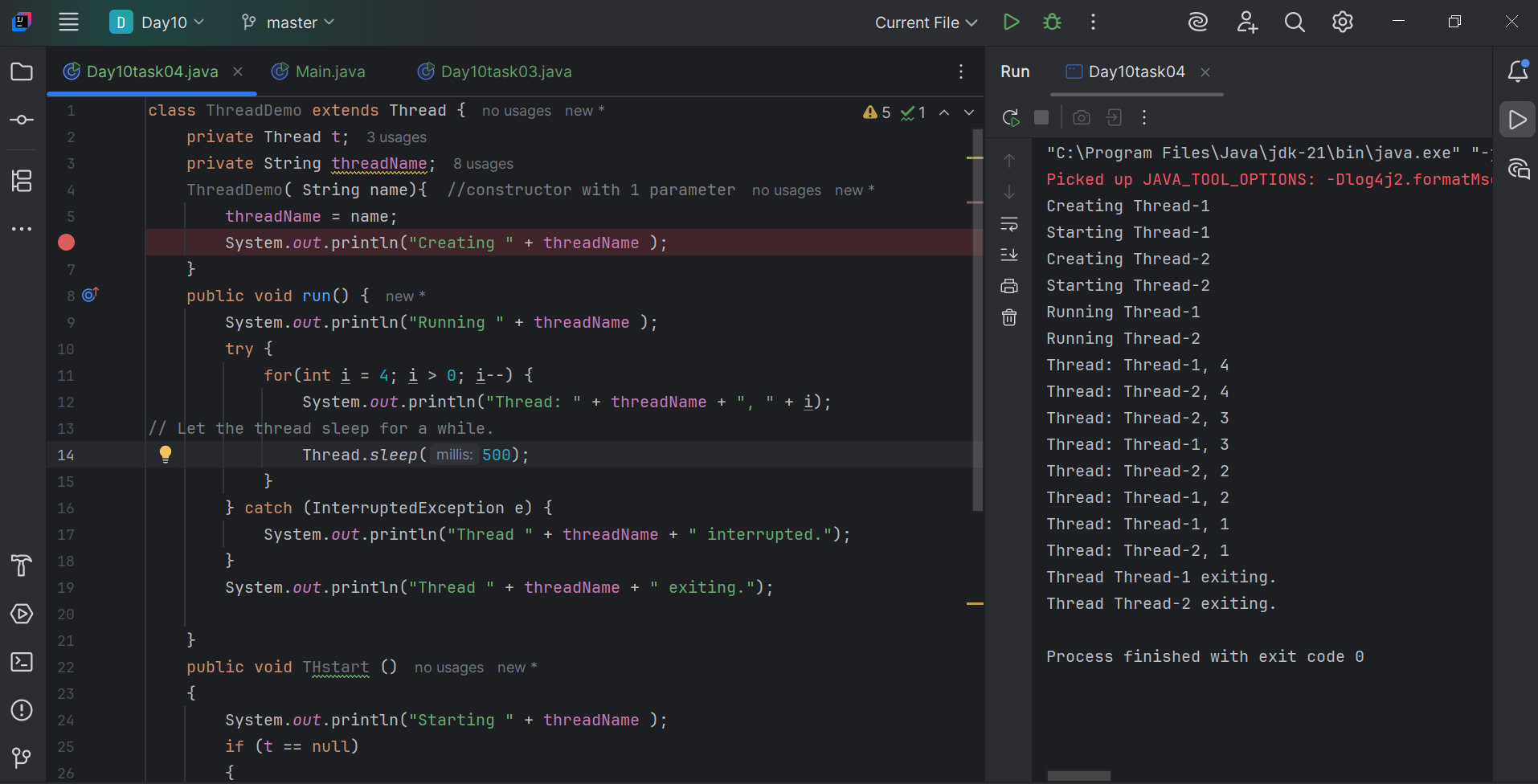
1. One thread shows the webpage.
2. Another thread plays a video.
3. Another thread checks for new messages.

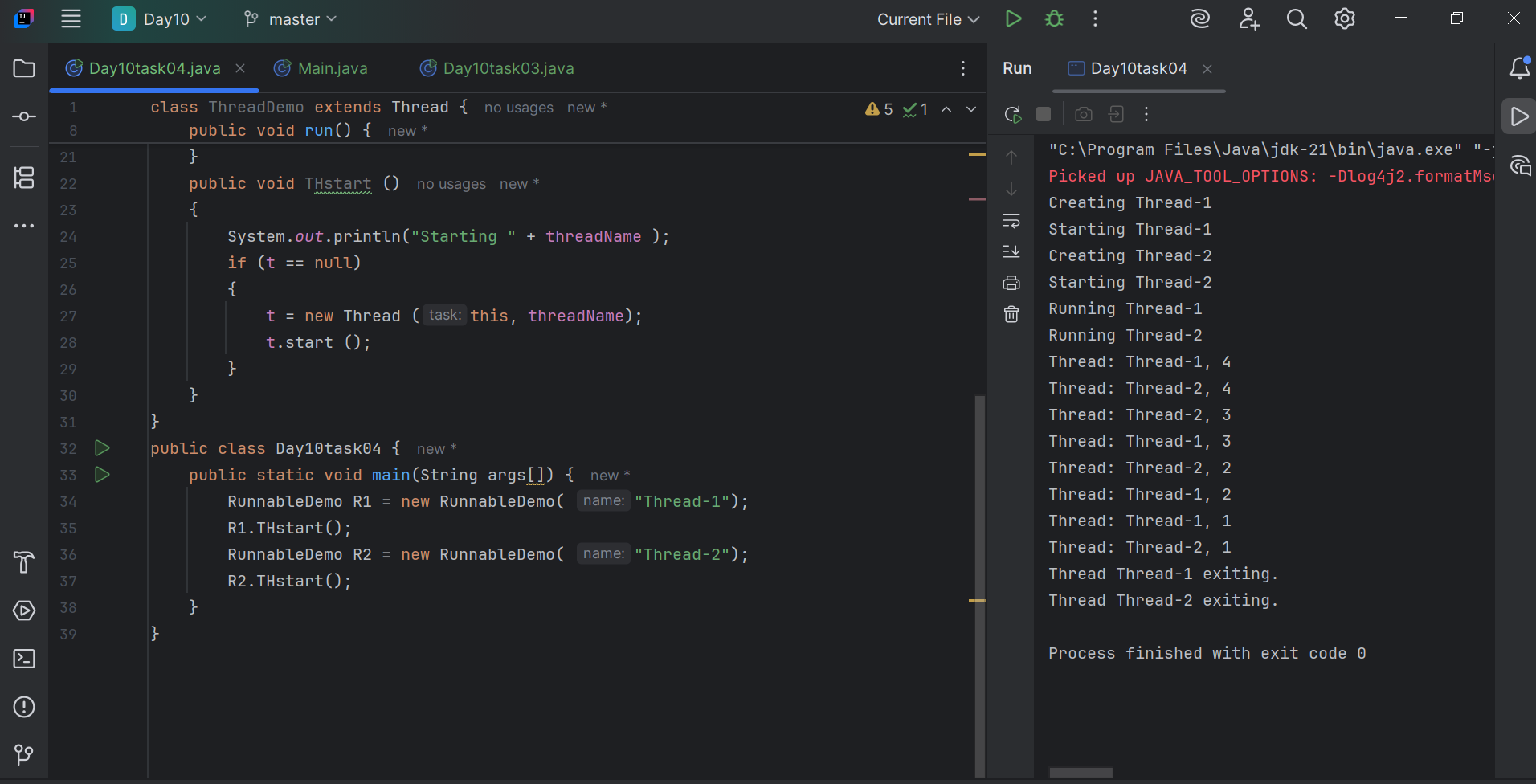
**Task - 3:**

****

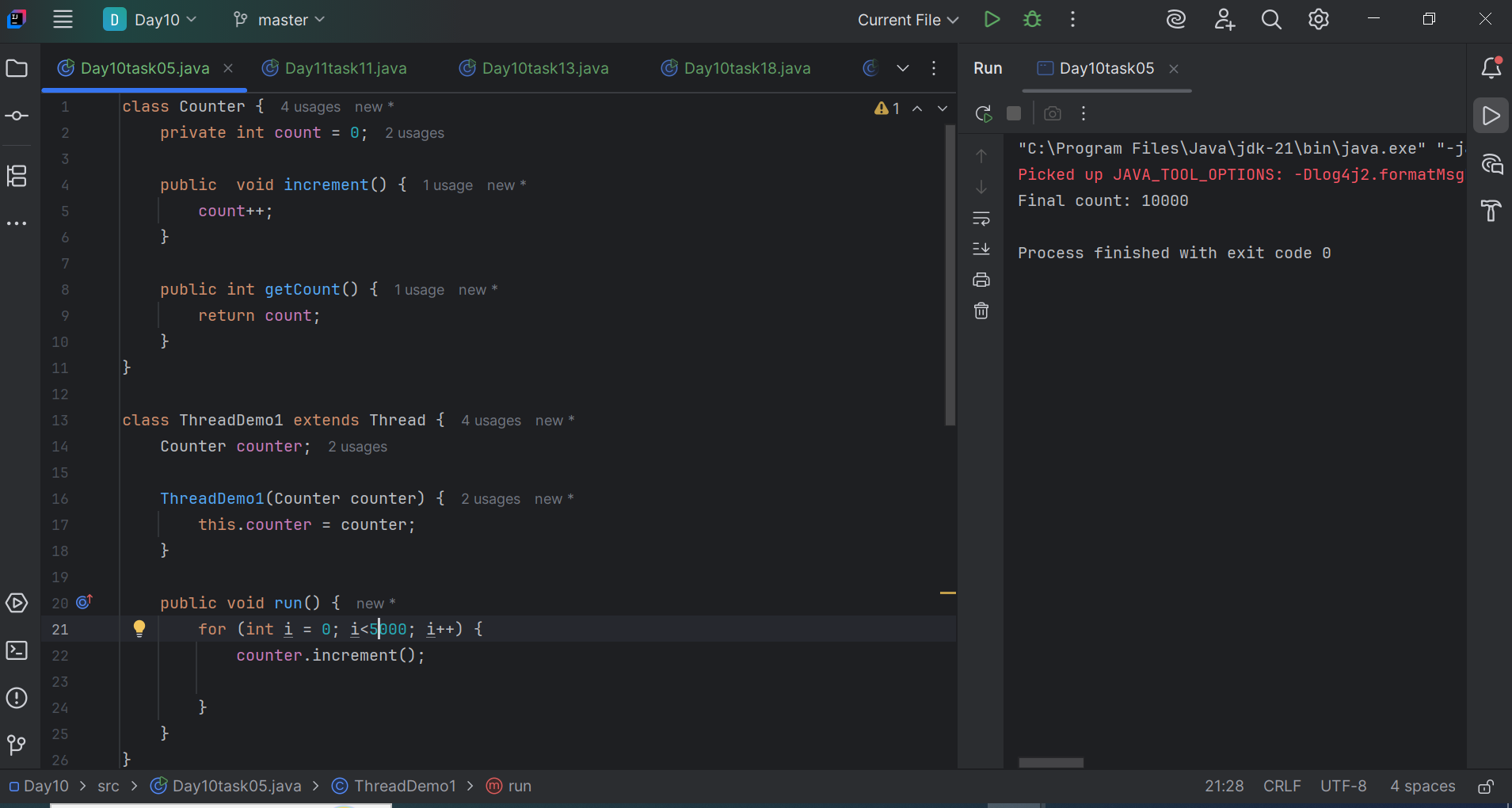
****

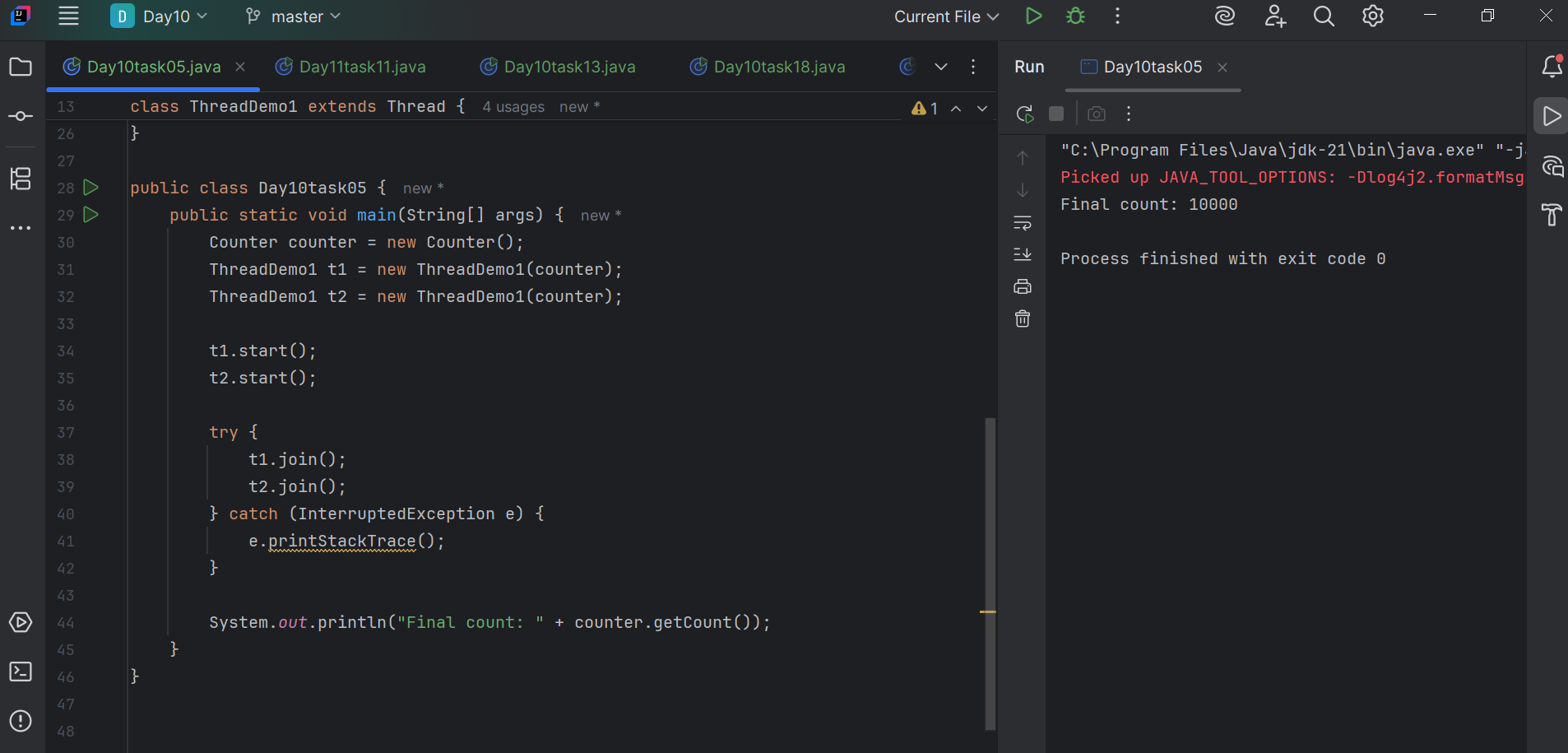
**Task - 4:**

****

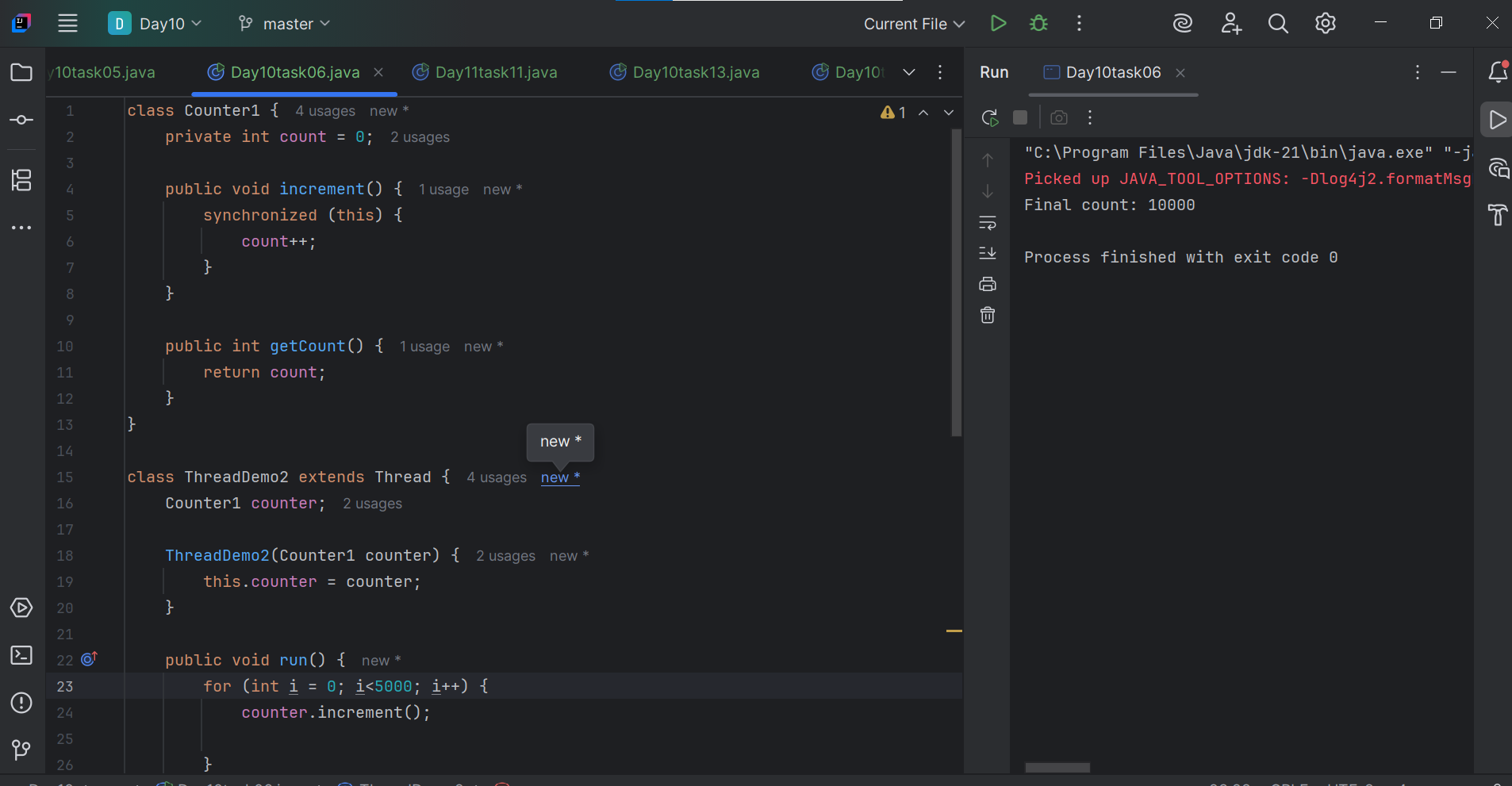
****

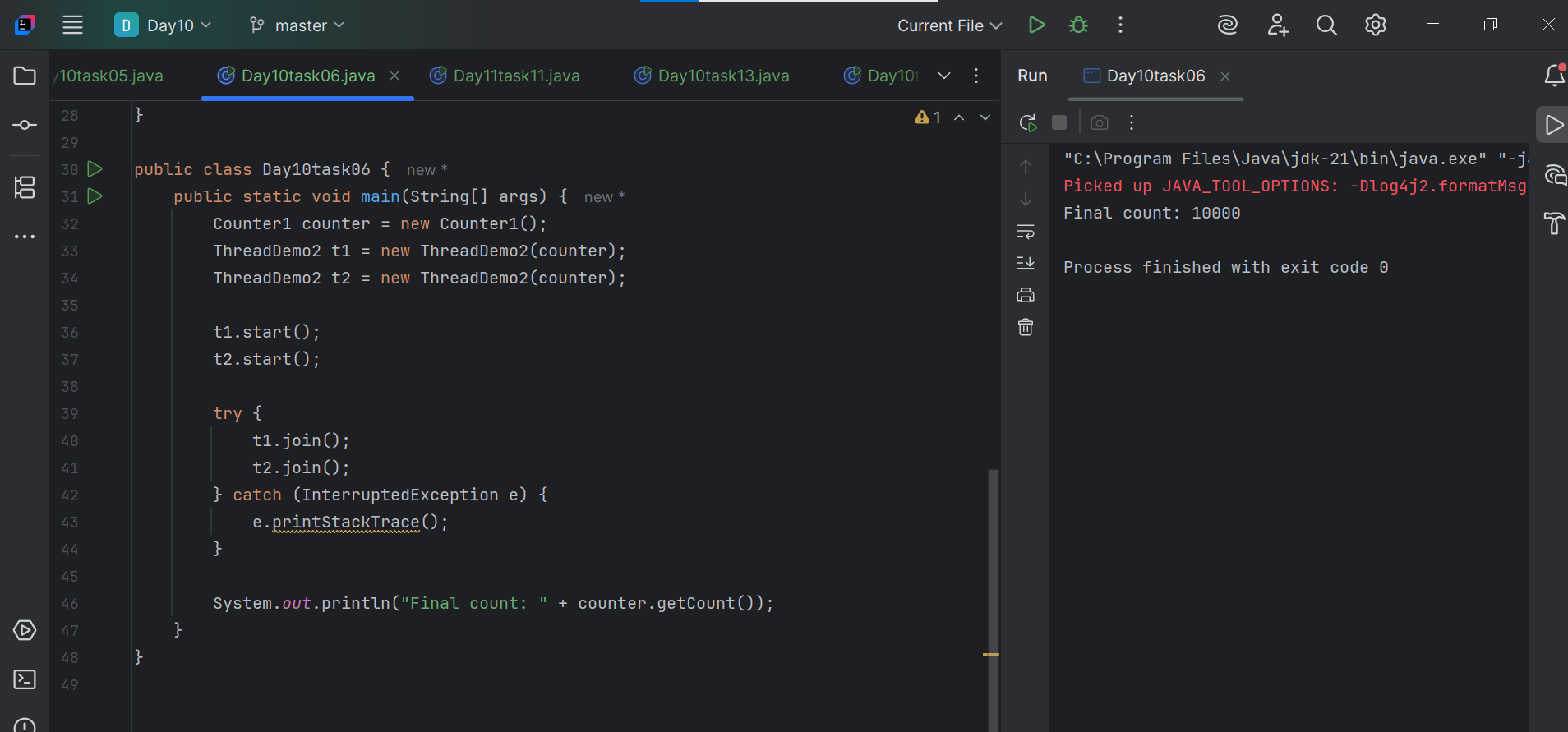
**Task - 5:**

****

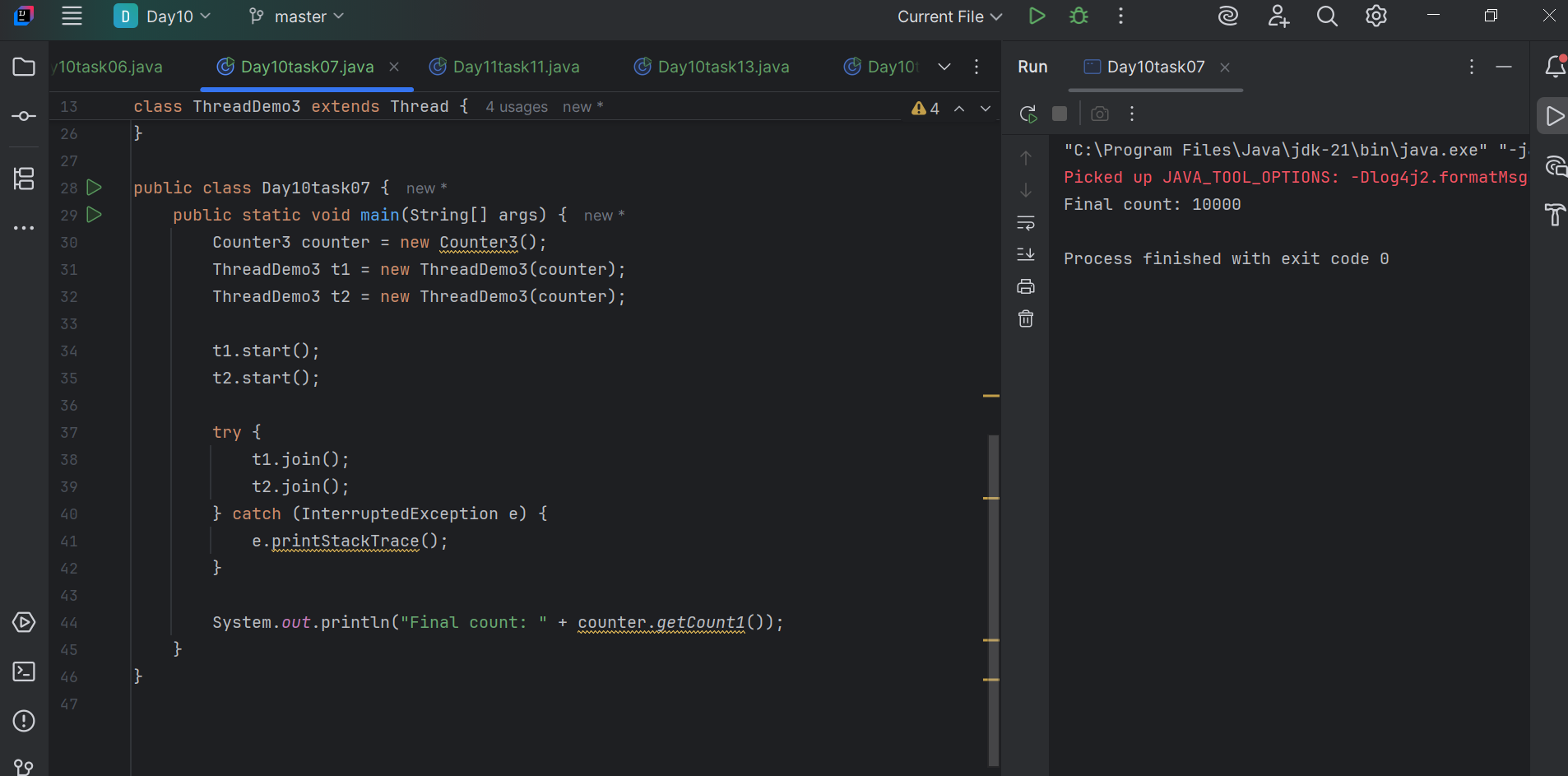
****

**Task -6:**

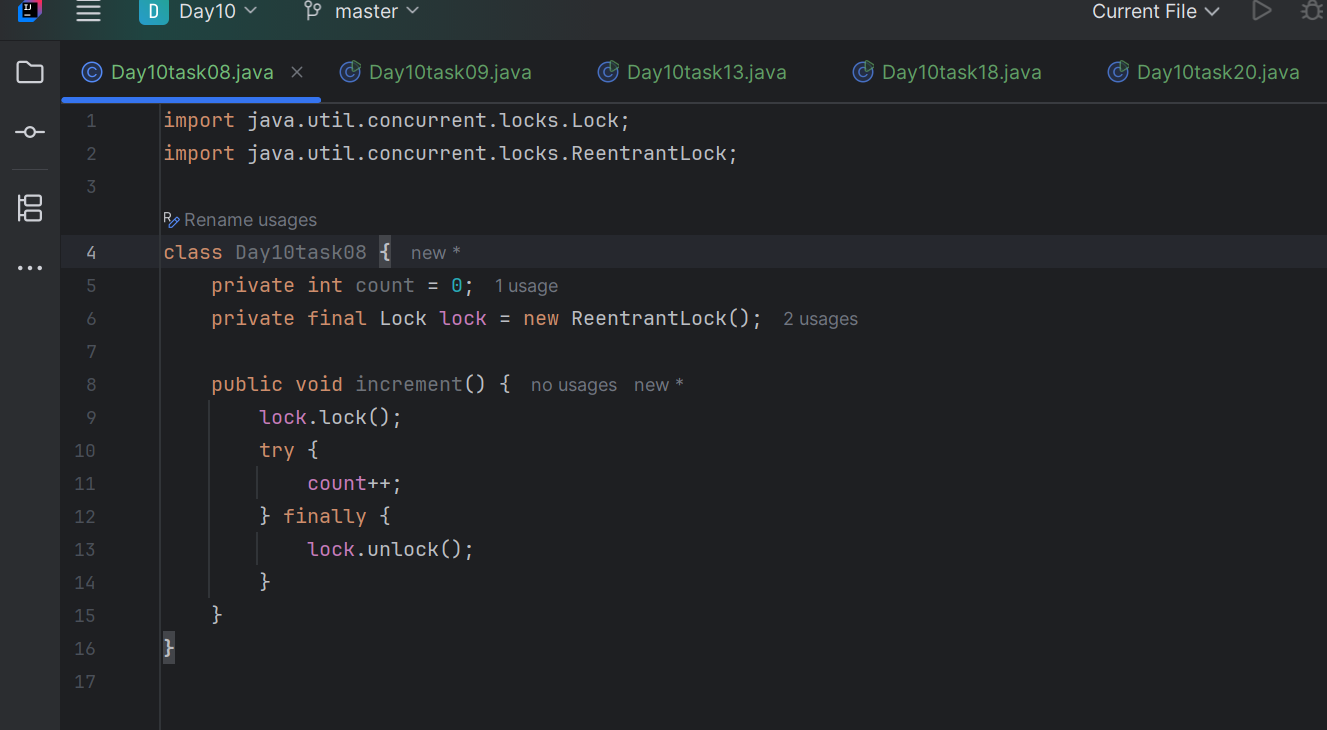
****

****

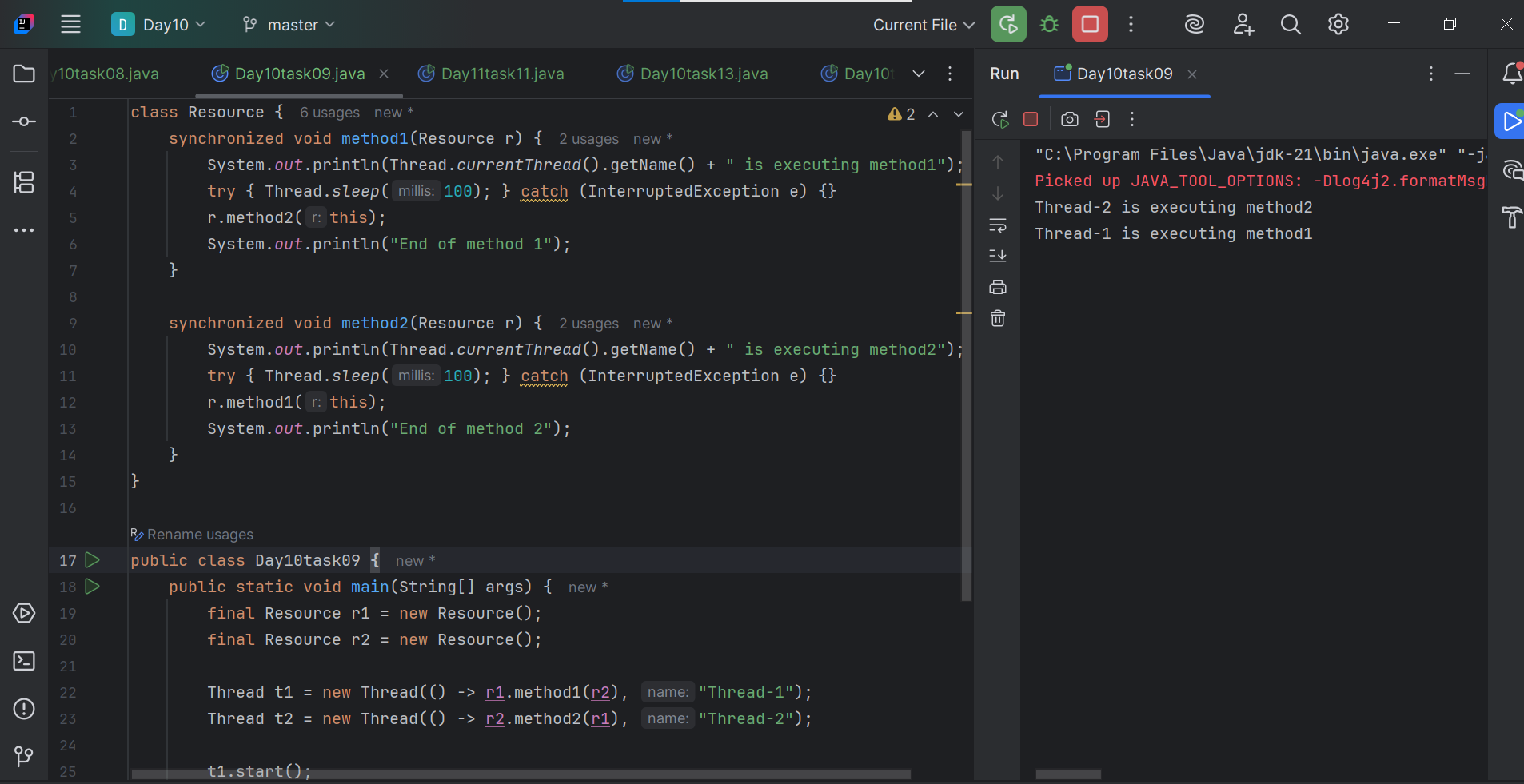
**Task -7:**

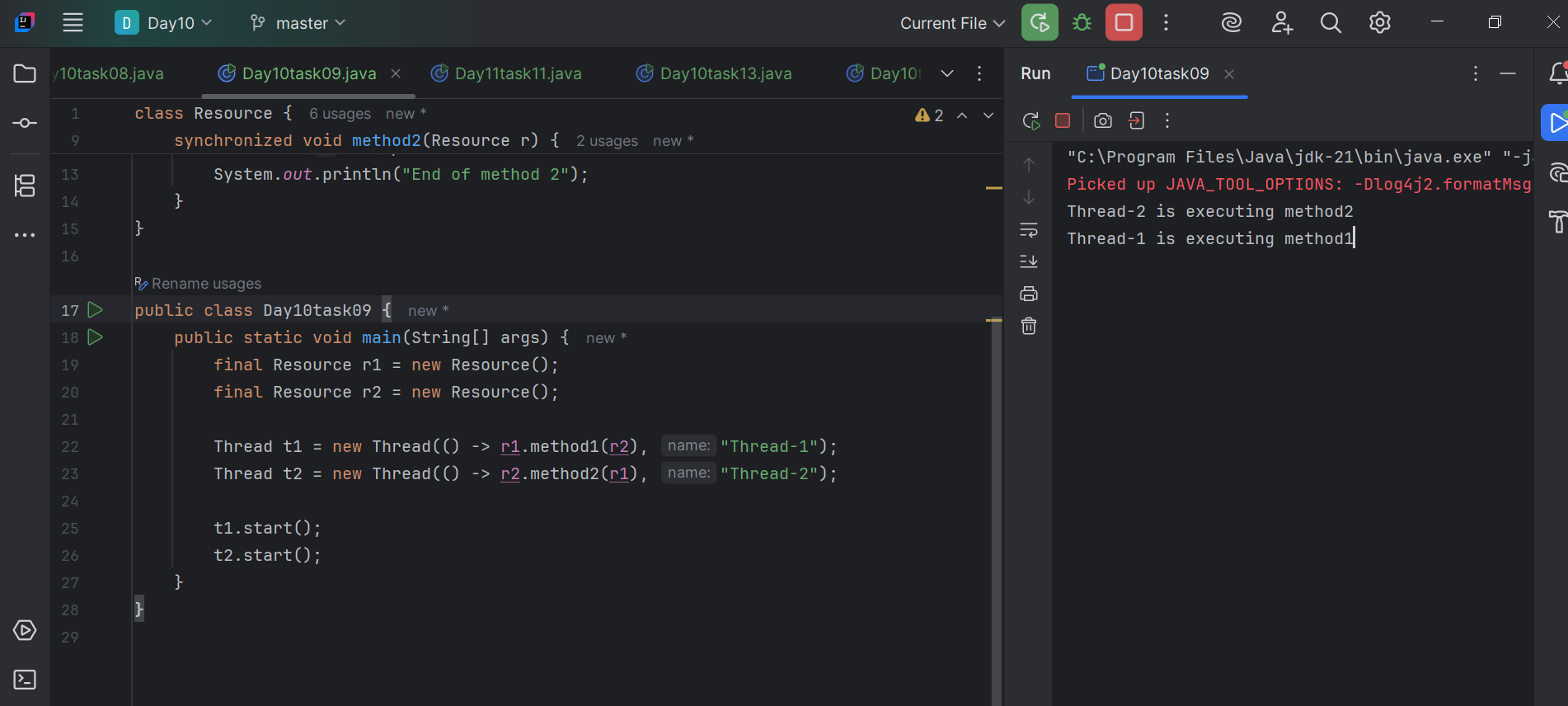
****

**Task - 8:**

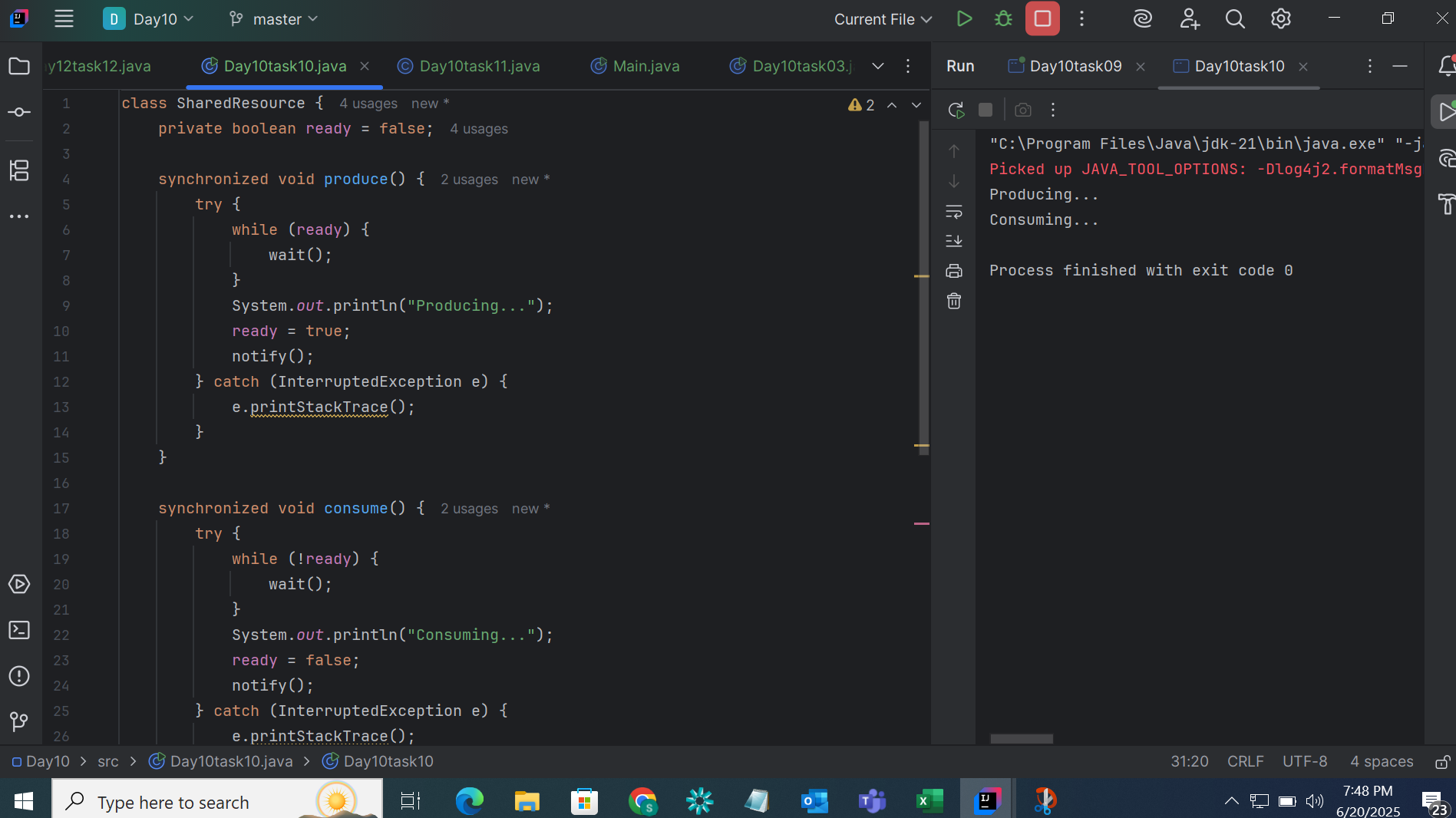
****

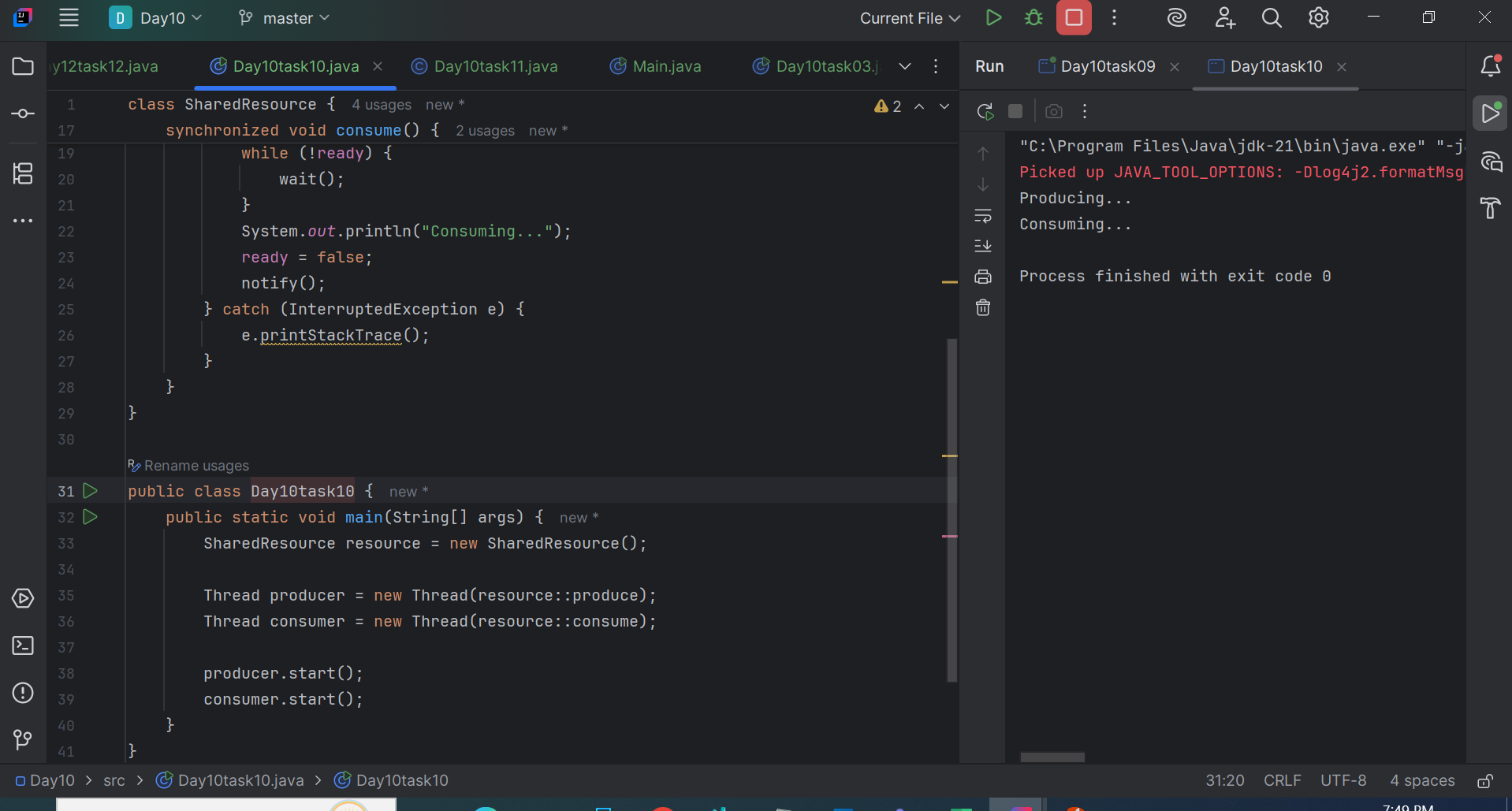
**Task - 9:**

****

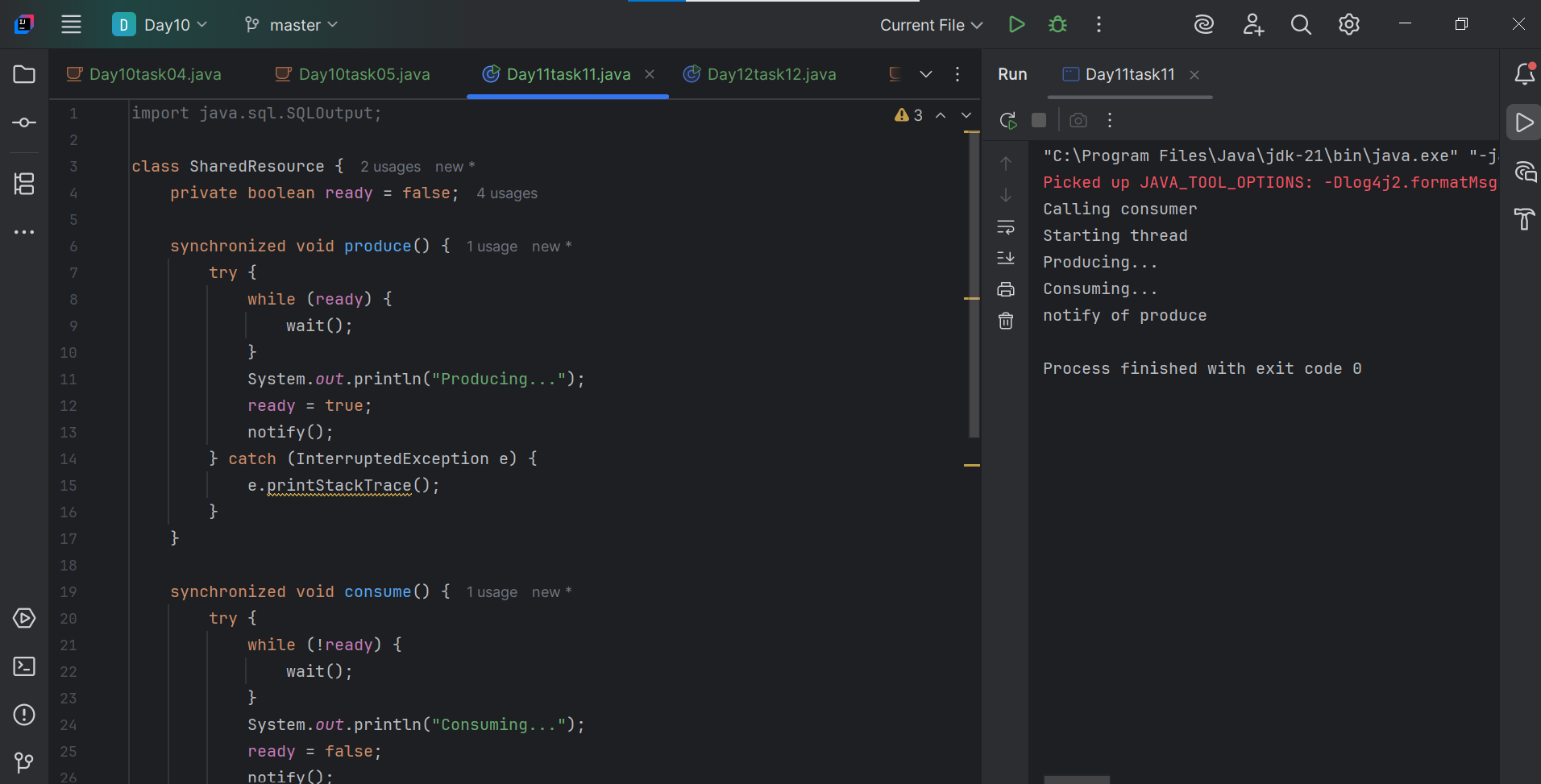
****

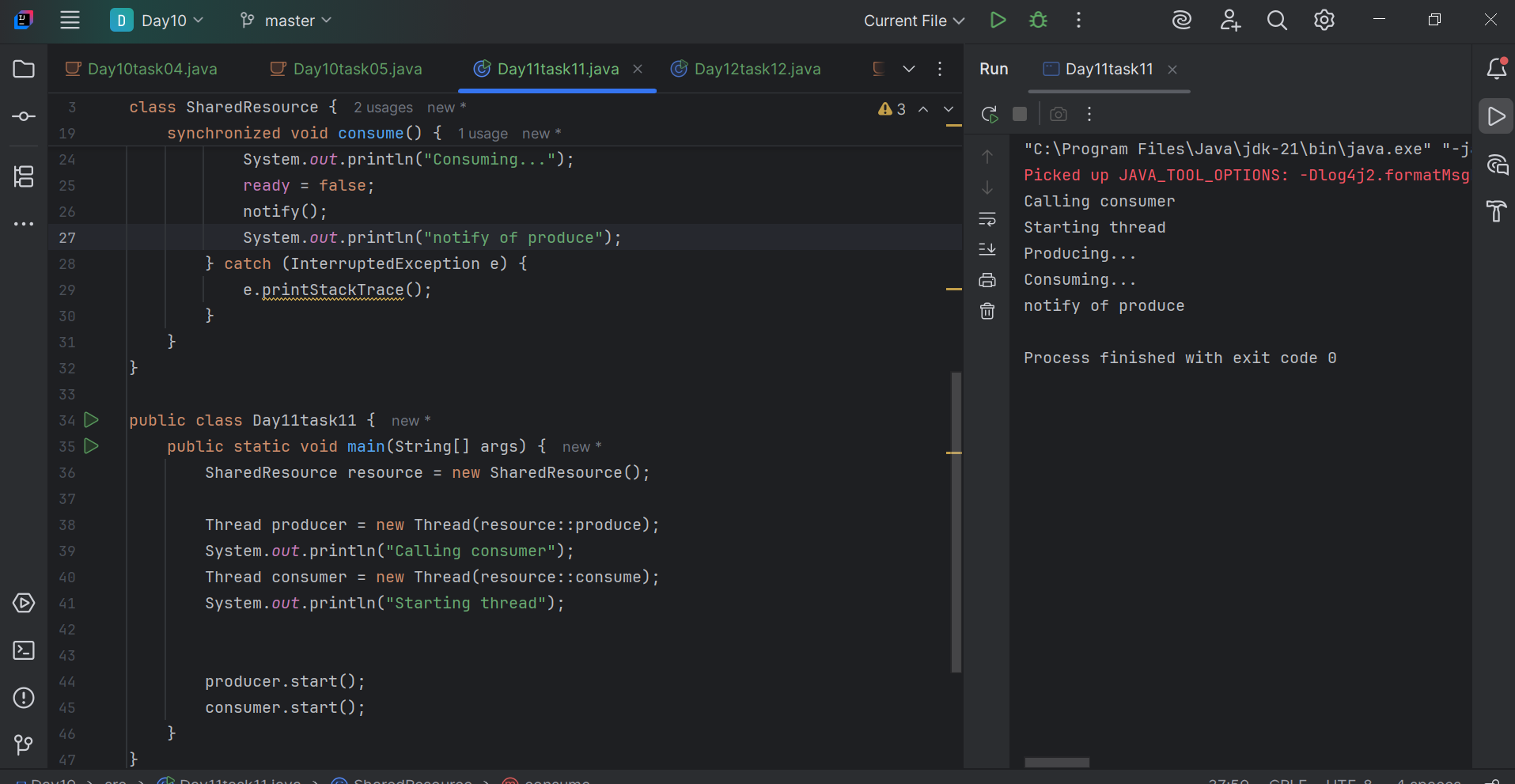
**Task - 10:**

****

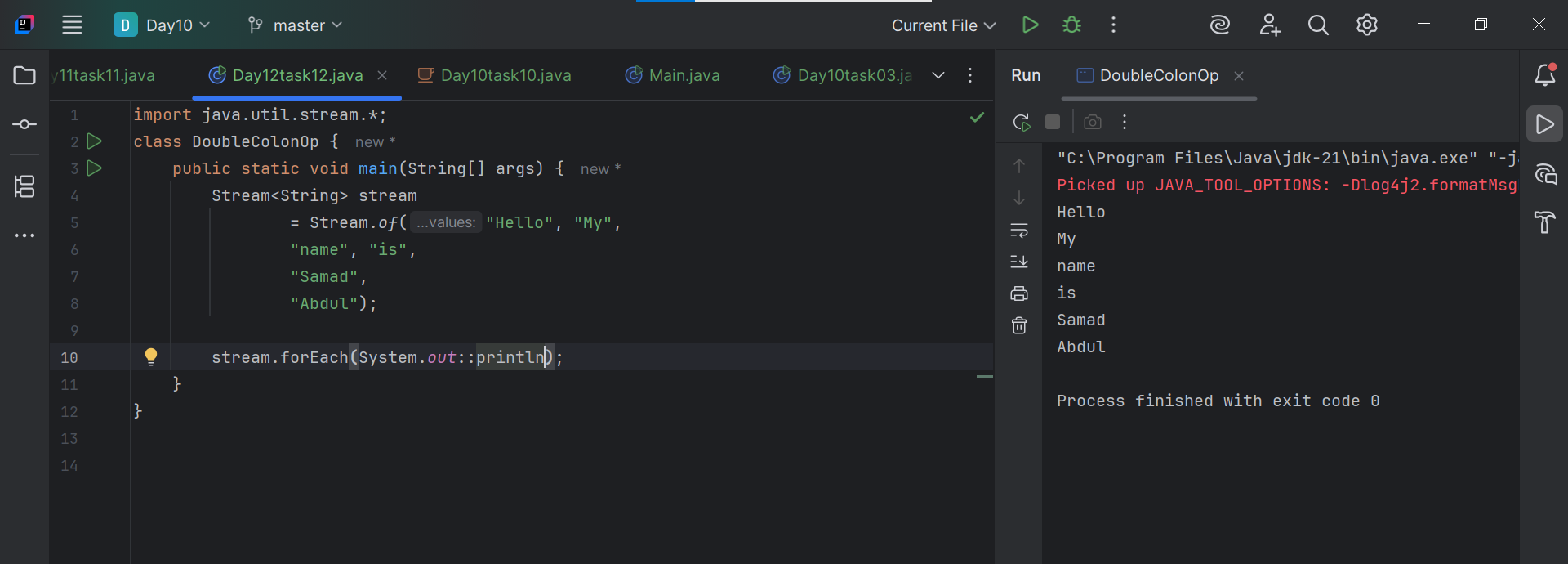
****

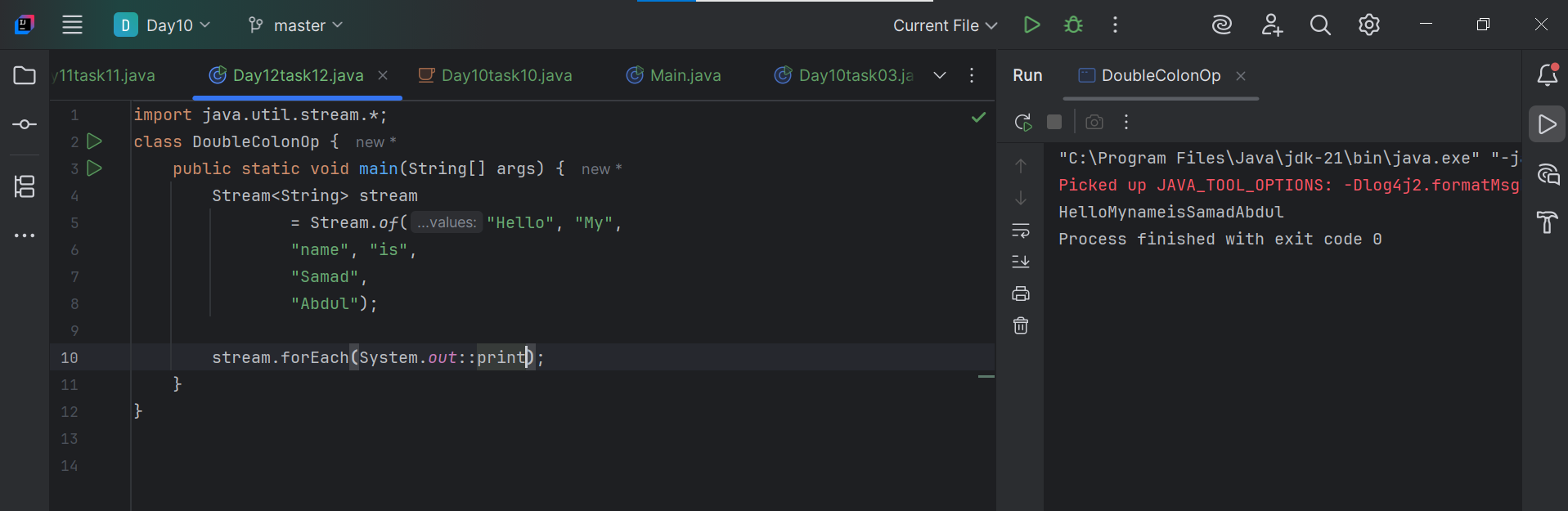
**Task -11:**

****

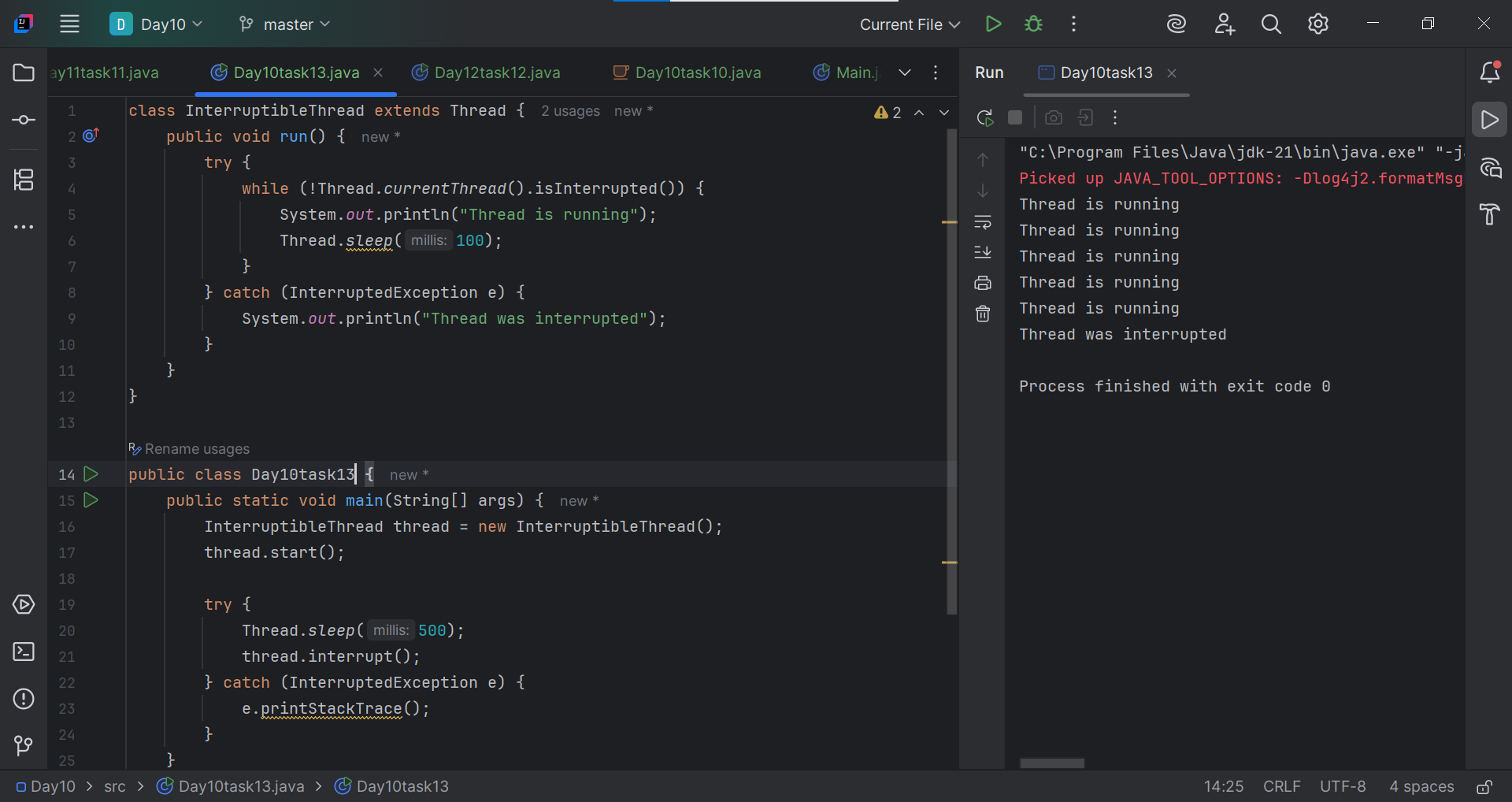
****

**Task - 12:**

****

****

**Task -13:**

****

**Task -14:**

**What are Daemon threads? Explain…**

A daemon thread is a background thread that runs behind the scenes to support other threads.

It automatically ends when all non-daemon (main) threads are finished.

### Example in Real Life:

* You're working on your computer (main thread).
* Antivirus software (daemon thread) is scanning in the background.
* Once you shut down your computer (main thread ends), the antivirus scan also stops automatically.

Also Daemon threads are used for:

* Garbage collection
* Monitoring
* Auto-saving
* If only daemon threads are left, the JVM exits — they don’t keep the program alive.

**Task -15:**

What are the debugging tools in Java.. list down a few..

| **Tool Name** | **Used For** | **Language/Platform** |
| --- | --- | --- |
| Visual Studio Debugger | Windows apps, Web apps | C#, C++, .NET, etc. |
| Eclipse Debugger | Java development | Java |
| IntelliJ IDEA Debugger | Advanced Java & Kotlin debugging | Java, Kotlin |
| Android Studio Debugger | Mobile apps | Java, Kotlin (Android) |
| Chrome DevTools | Web development debugging | JavaScript, HTML, CSS |
| Xcode Debugger | macOS & iOS apps | Swift, Objective-C |
| GDB (GNU Debugger) | Low-level debugging | C, C++ |

**Task -16:**

**What are the error Messages.. What are they and when do we see them?**

Error messages are short pieces of text that a program shows when something goes wrong.  
They tell us what happened, where, and often why, so you—or the user—can fix or avoid the problem.

## Why do they exist?

|  |  |
| --- | --- |
| Alert | Let the user or developer know *something failed or is about to*. |
| Diagnose | Provide clues (file name, line number, error code) so the issue can be traced and fixed. |
| Guide | Suggest next steps—retry, check settings, contact support, etc. |
| Protect | Stop the program before it corrupts data or crashes harder. |

## When do we see them?

|  |  |
| --- | --- |
| Compile time | Syntax error, missing semicolon, wrong type (“cannot convert int to String”). |
| Start‑up | Configuration file missing, port already in use. |
| Run time | Division by zero, file not found, network unreachable, NullPointerException. |
| Shutdown / cleanup | “Resource leak detected,” “Database connection still open.” |

**Task - 17:**

**What is Stack trace.. What will it do?**

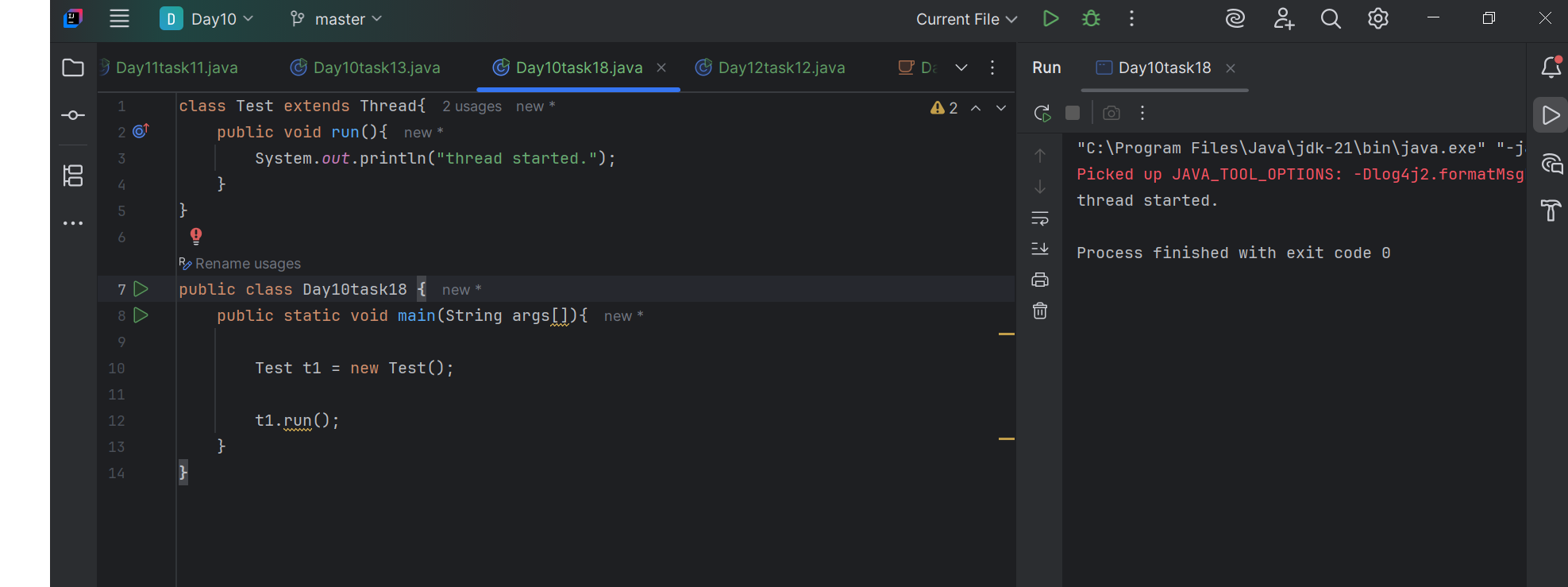
A stack trace is a list of method calls that shows where your program crashed or had an error.

It helps you trace back the steps your code took just before the problem happened.

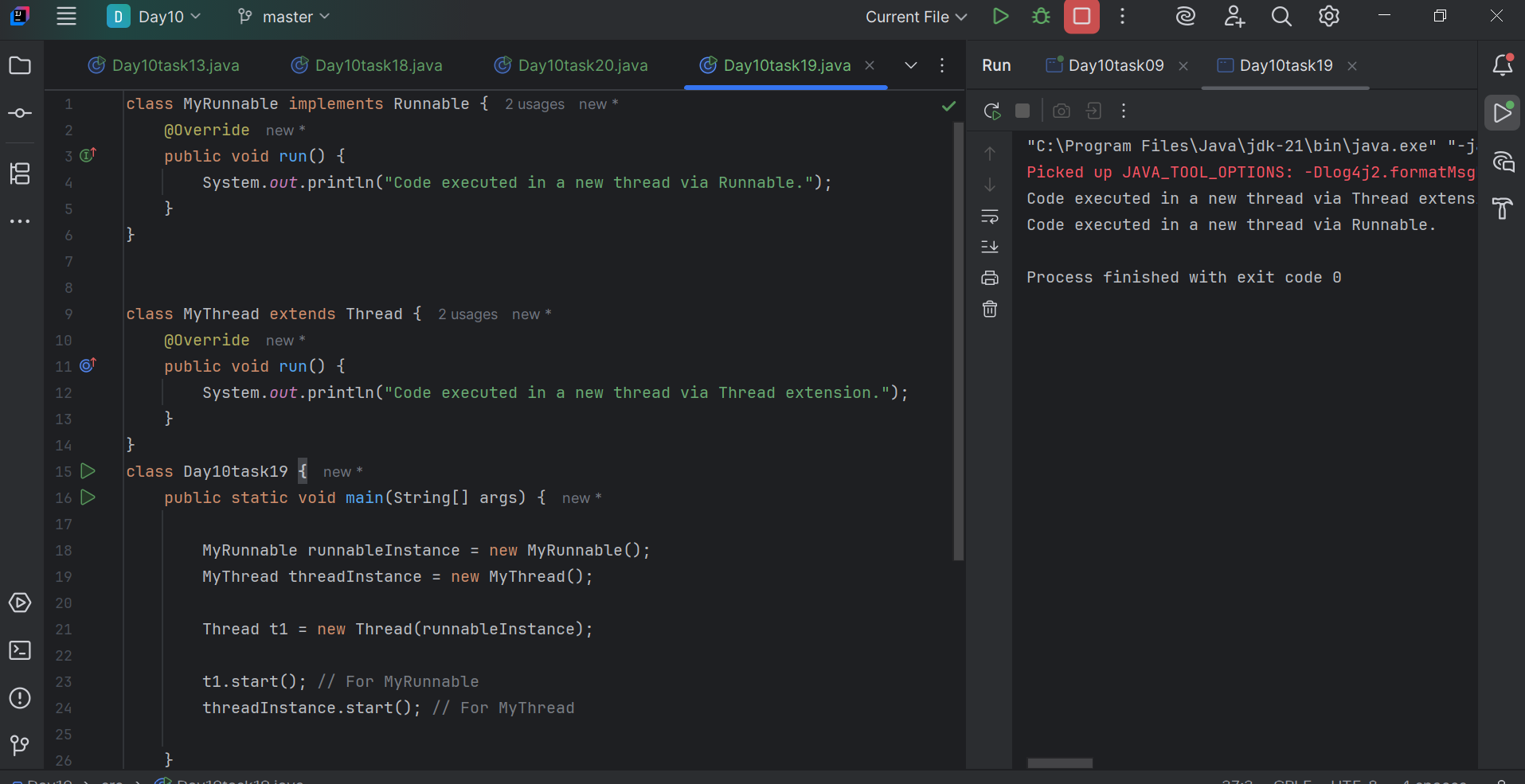
A stack trace helps us to:

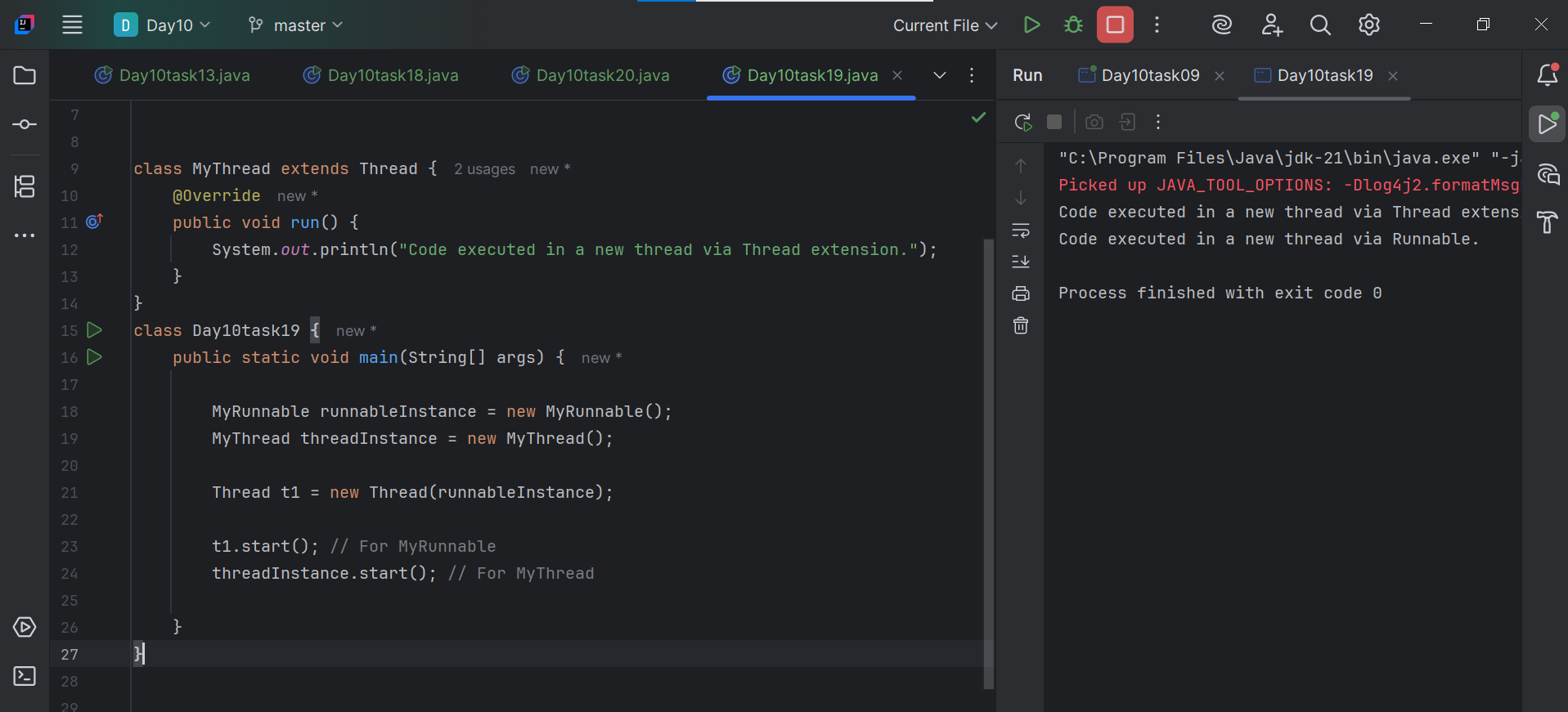
1. Identify the error – What type of exception happened? (e.g. NullPointerException, ArrayIndexOutOfBoundsException)
2. Locate the code – Which line in which file caused the error?
3. Analyze the path – Which methods were called before the error?
4. Fix the problem – So it doesn't happen again.

**Task -18:**

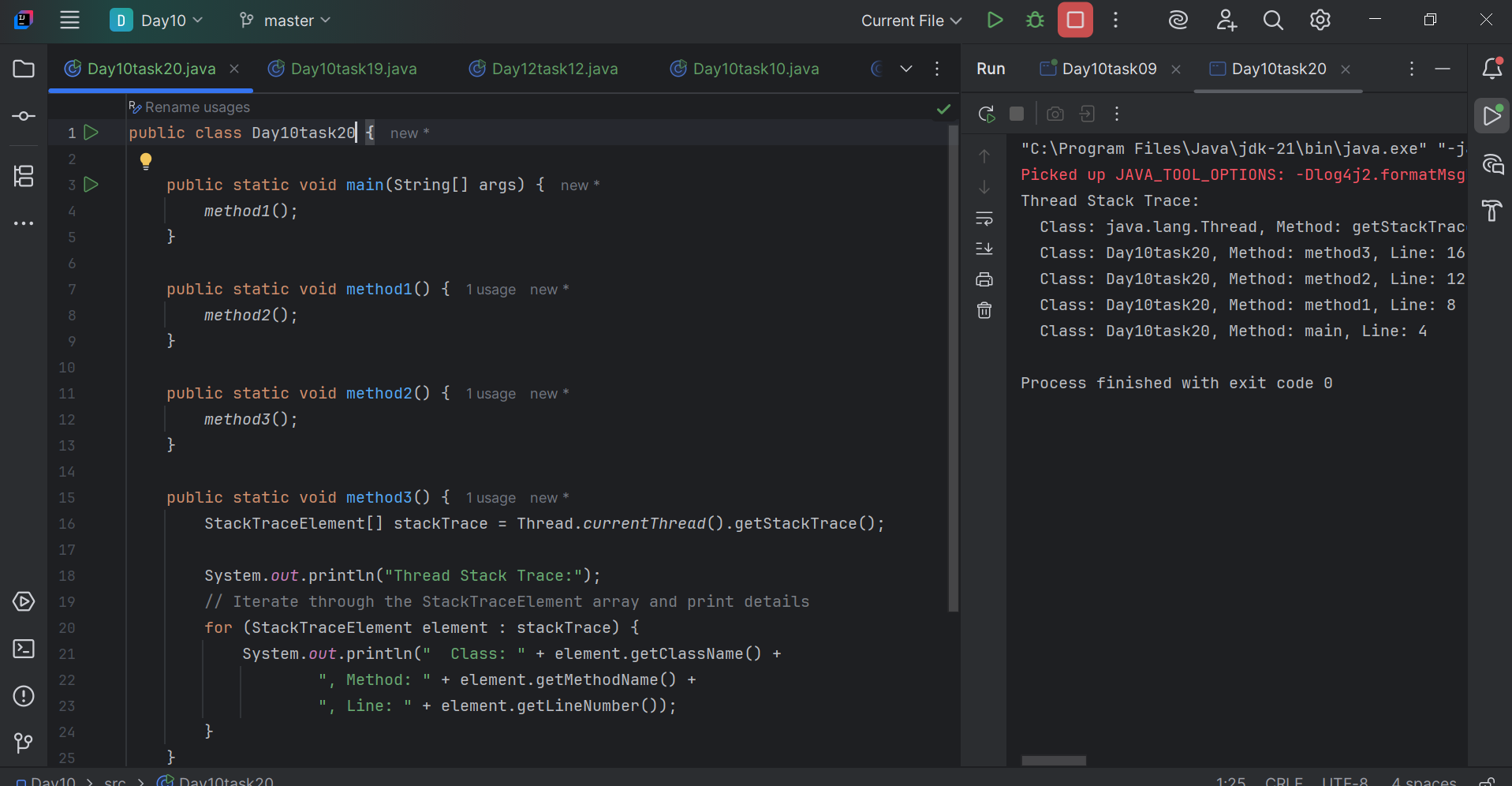


**Task -19:**

****

****

**Task -20:**

****