Day 10 - 104608492 - Shirisha Perapagu

Multithreading

Task 01

What is a process?

A process is an independent program running or simply an active program.

Processes are isolated from each other and has its own memory space.

Example:

If we open an application and use WhatsAapp then it is aprocess.

Similarly, if we open any browser and use any site, it is a process.

Task 02

What is a thread?

A thread is a small task running inside a process.

There can be multiple threads inside a process which allows us to do multiple things at once.

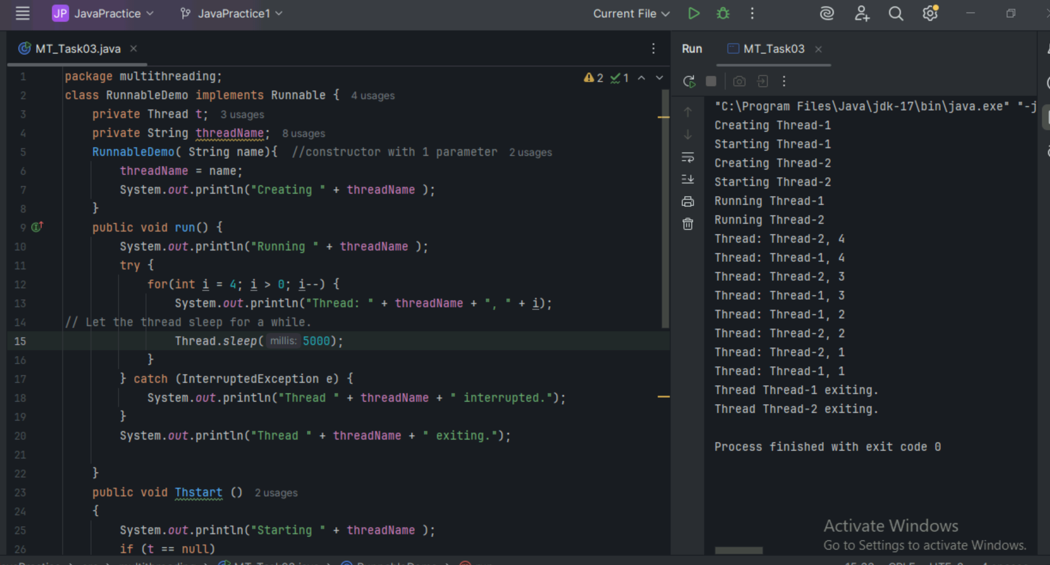
All threads share same memory space.

Example:

For a WhatsApp application, one thread can be user receiving messages, another thread can be user is on voice/video call, another thread can be user sending the message and so on.. allowing users to do multiple tasks at a time.

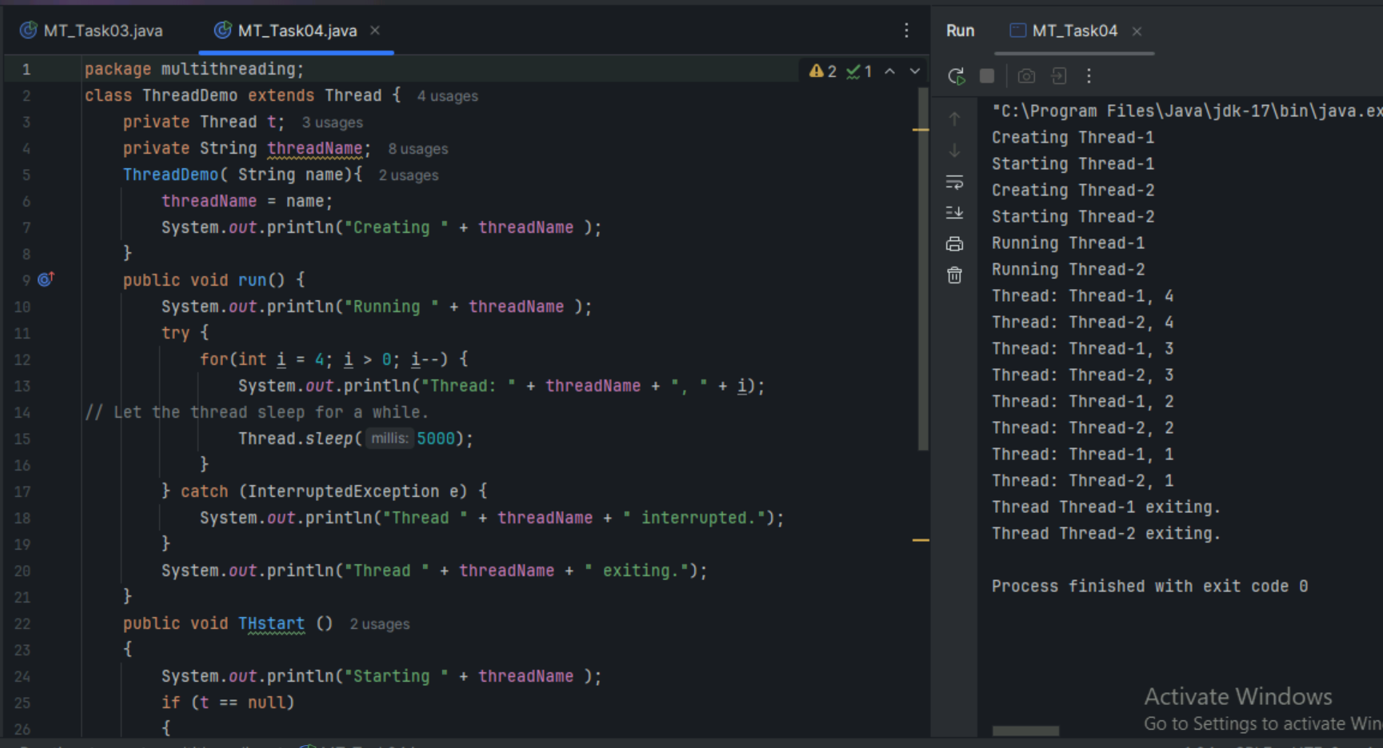
Task 03

Thread creation by Implementing Runnable Interface



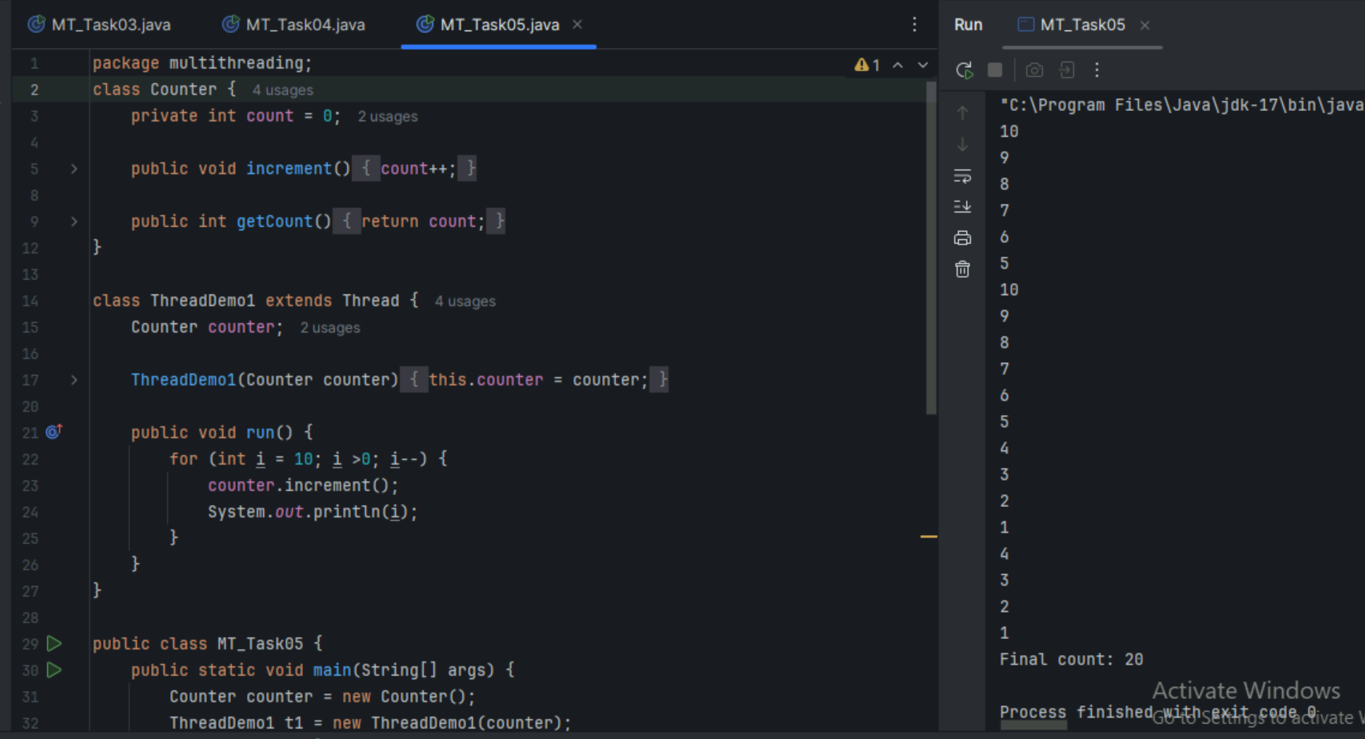
Task 4

Thread creation by extending Thread class



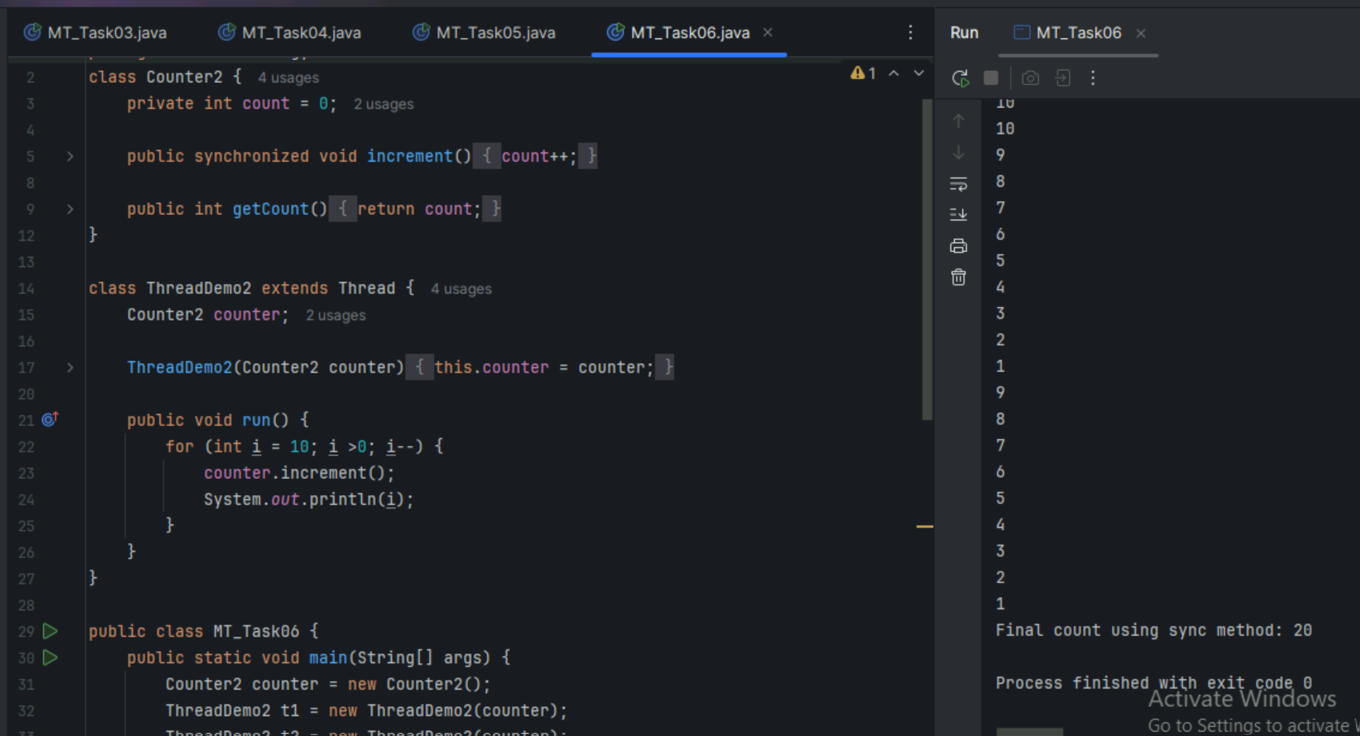
Task 05

Check the output without synchronized method



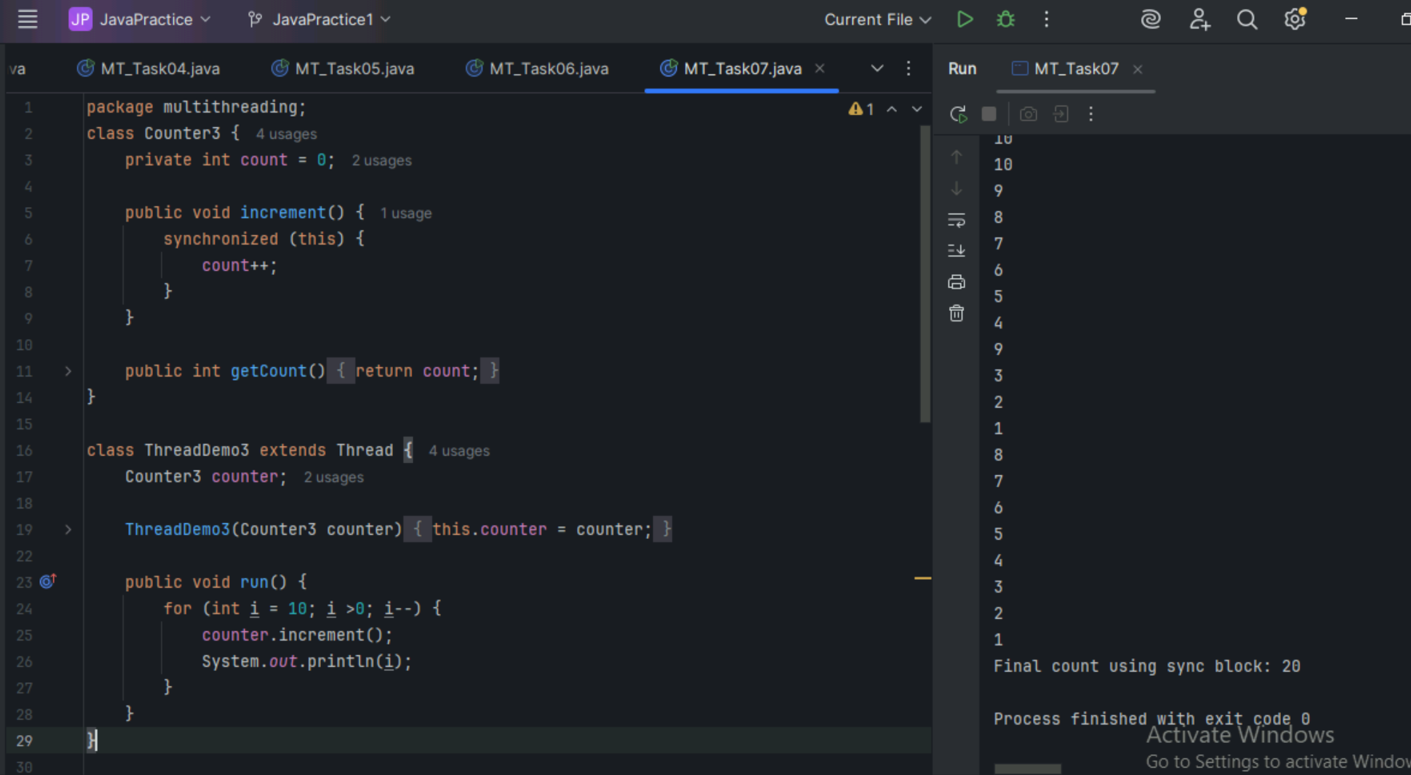
Task 06

Check the output with synchronized method



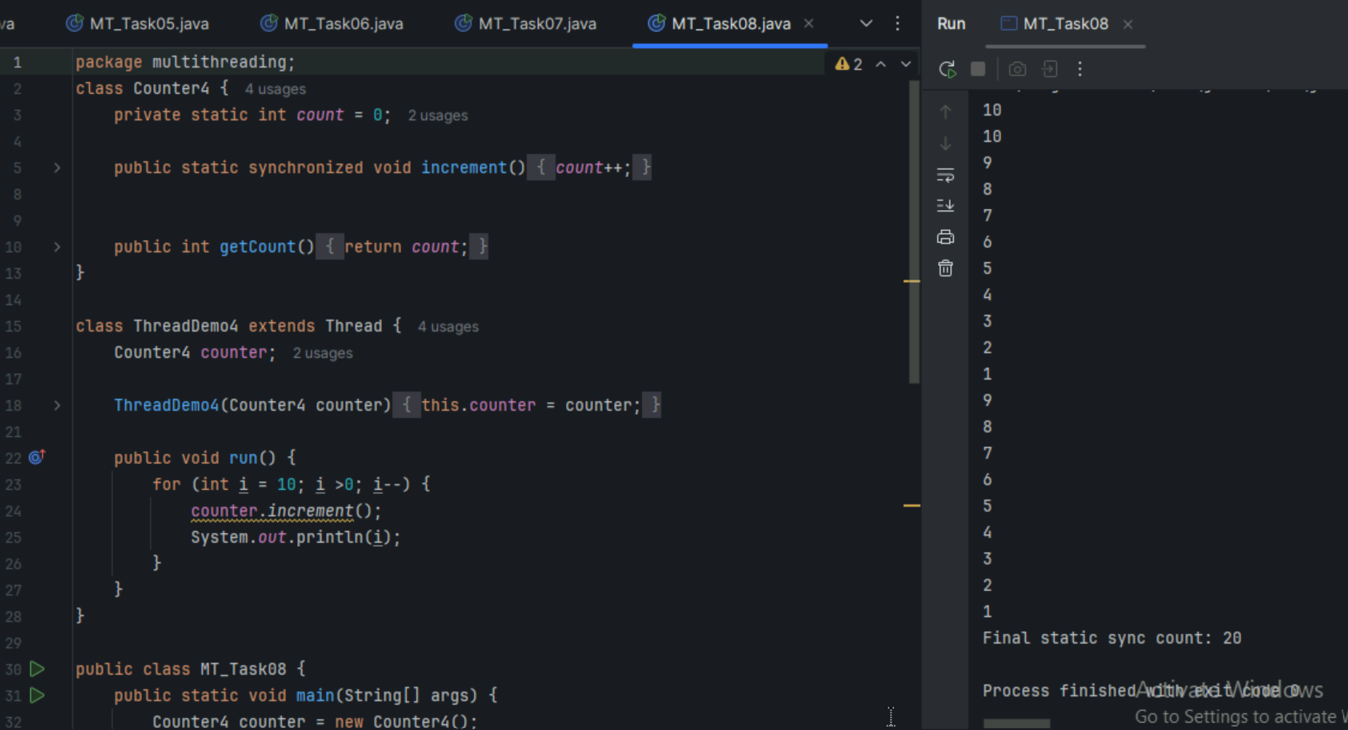
Task 07

Check the output using synchronized block



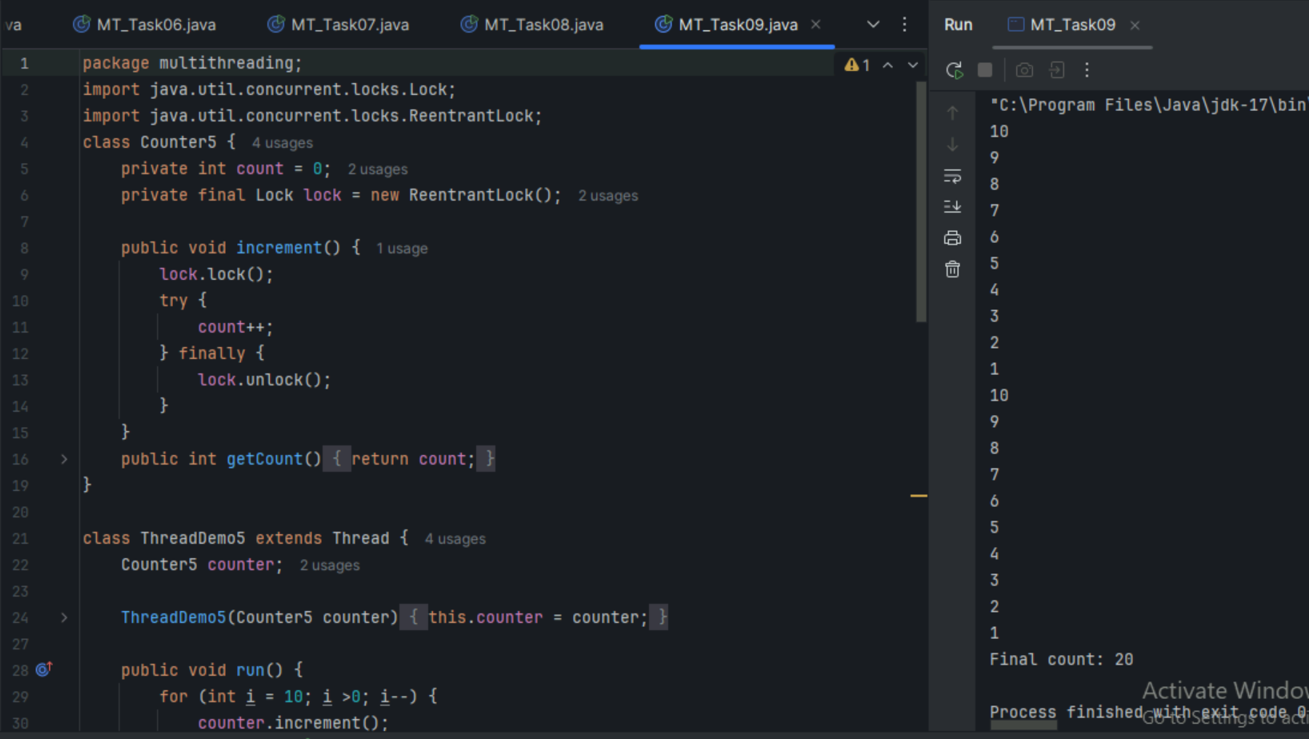
Task 08

Check the output using static synchronized method

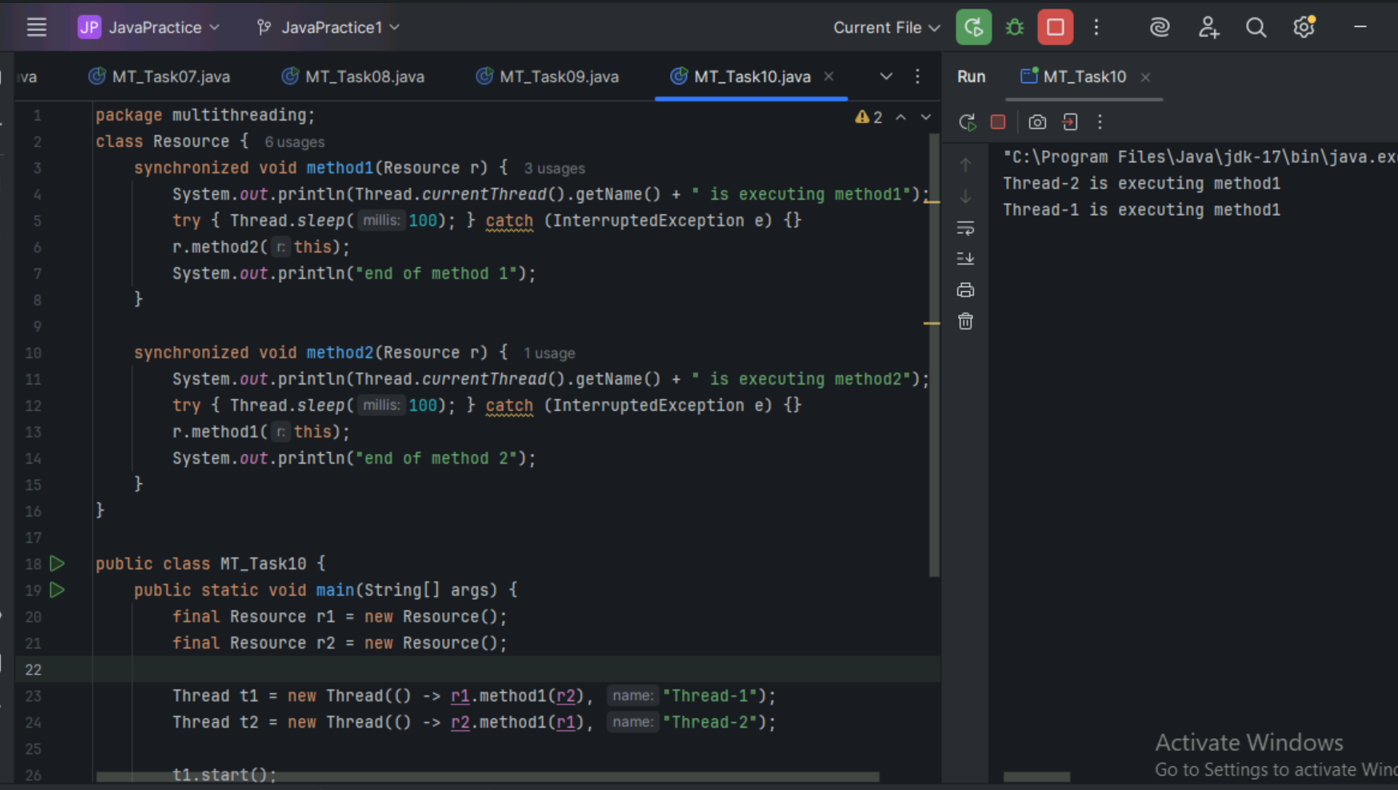


Task 09

Wap to demonstrate Locks.

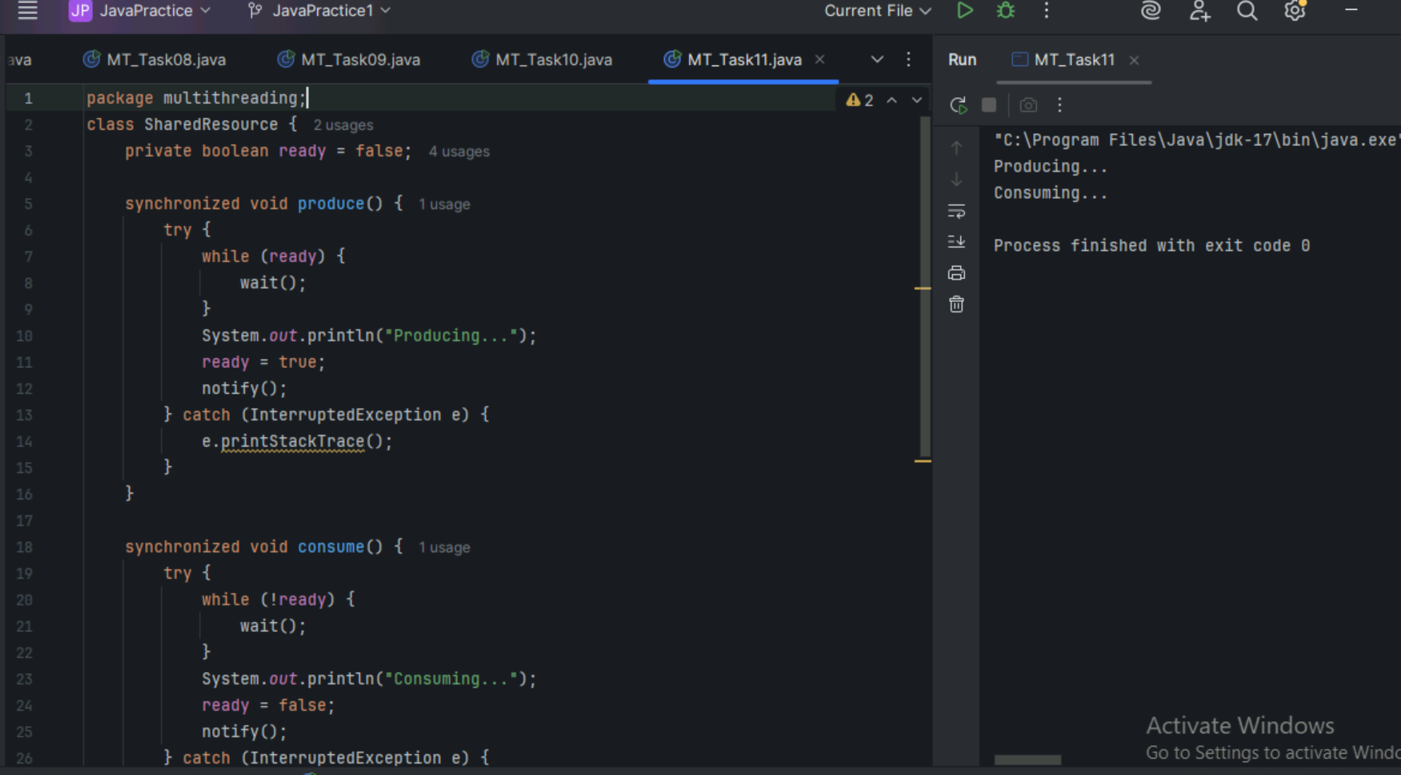


Task 10 : Wap to demonstrate Deadlock situation.



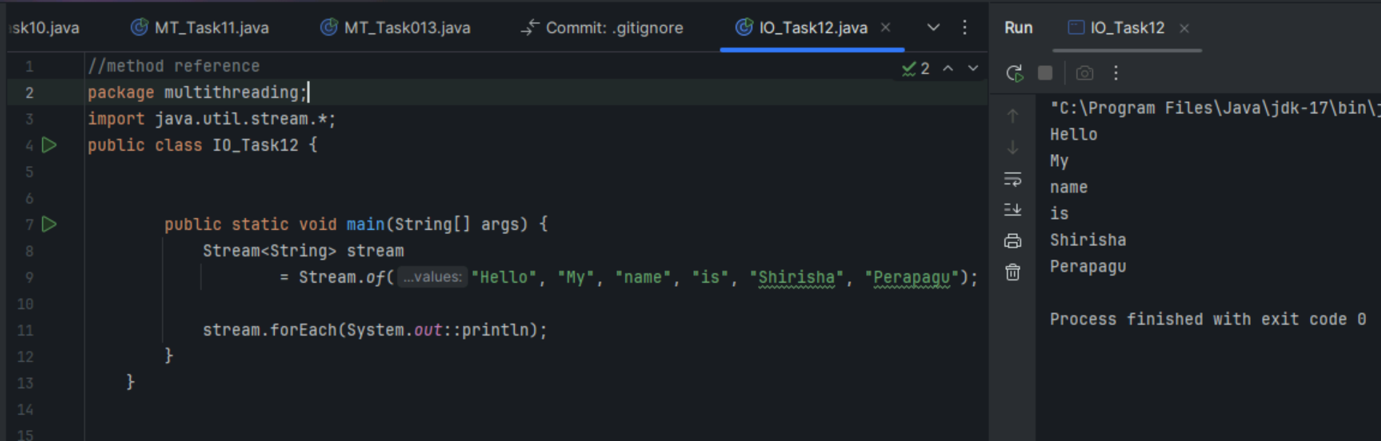
Task 11:

Wap to demonstrate Inter-thread communication.



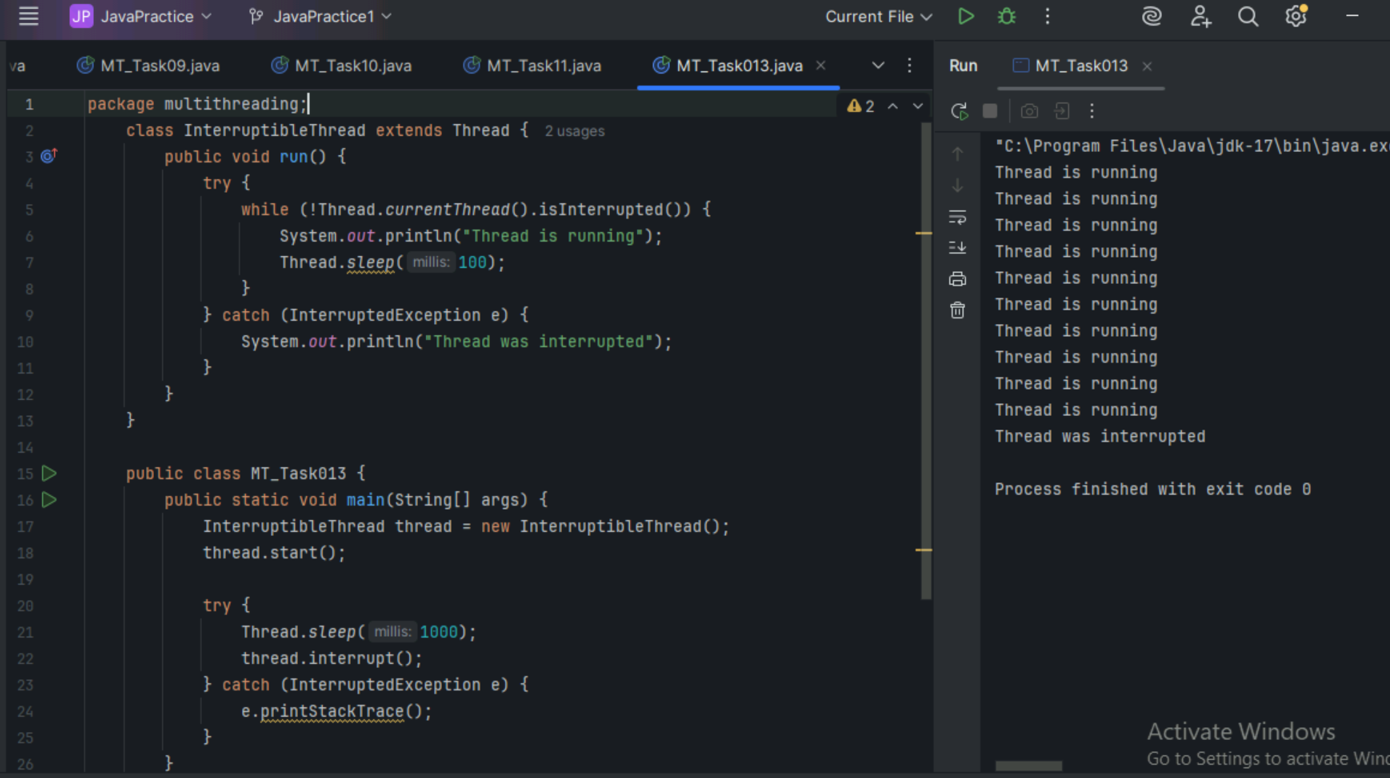
Task 12:

Stream example



Task 13

Example of Interrupting a Thread



Task 14

What is Daemon Thread?

It is a background thread that runs in the background to support other main threads.

It automatically stops when all the main threads work is finished.

For Example:

In WhatsApp application.

Main thread can be main work like chatting.

Daemon thread helps in the background like checking for new messages from the server.

Daemon thread stops when main work is over.

In Java program, some of daemon threads are

1. Garbage Collector – used to free up memory by removing unused objects. This runs in background and when the program ends it will also get ended.
2. Finalizer Thread – used to clean up resources like closing files etc. before an object is removed. JVM kills daemon thread automatically once all main threads are done.

Task 15

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

1.Eclipse

2.IntelliJ IDEA

3.Visual Studio Code

4.Netbeans

Task 16

Try to understand the error Messages.. What are they and when to use?

Compile Time Errors:

1. Syntax Errors

Eg. Missing semicolon, braces, parenthesis etc.

Int a

Int b;

**Error: ‘;’ expected**

1. Type Mismatch

Eg. Assigning incompatible types

Int x=”text”;

**Error: incompatible types: String cannot be converted to int**

1. Cannot Find Symbol

When we use a variable or method which is not been declared.

Eg.

Int num1=5,num2=10;

Int total=num1+num2;

System.out.println(sum);

**Error: cannot find symbol**

**Symbol : variable sum**

**Location: class Main**

1. Unreachable code

Code after return statement in a method or inside if(false) block.

Eg1.

public void getValue()

{

return;

System.out.println(“This line is unreachable code”);

}

Eg2.

if(false){

System.out.println(“This block is unreachable code”);

}

**Error: unreachable statement**

1. Variable might not have been initialized

Java does not give default value for local variable if not assigned, so we must need to assign value before using the variable.

Public static void main(String[] args){

Int x;

System.out.println(x);

}

**Error: variable x might not have been initialized**

Run Time Errors or Exceptions:

1. NullPointerException

Thrown when we try to use a null object reference.

Eg.

String str=null;

System.out.println(str.length());

**Exception:….java.lang.NullPointerException: Cannot invoke “String.length()” because “str” is null…**

1. ArrayIndexOutOfBoundsException

When we try to access an invalid index of array

Eg.

int[] arr={1,2,3};

System.out.println(arr[3]);

**Exception:….java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length**

1. ArithmeticException

When math operations are not valid like divide by zero.

int a=10,b=0;

System.out.println(a/b);

**Exception:….java.lang.ArrayIndexOutOfBoundsException: / by zero**

1. StackOverflowError

Caused by excessive deep recursion

Eg.

public static void recursive(){

recursive();

}

**Exception:….java.lang.StackOverflowError**

1. ClassCastException

When invalid casting is done between classes.

Eg.

String s=(String) new Integer(10);

**Exception:….java.lang.ClassCastException: class java.lang.Integer cannot be cast to class java.lang.String**

Task 17:

What is Stack trace.. What will it do?

When an exception or error occurs, stack trace gives the textual representation of the call stack at a moment.

it shows exact path of method calls that led to the error based on call stack at that moment.

Working steps:

1.JVM will start pushing method calls onto the call stack starting from main().

2.As methods call other methods each call is pushed into the stack in order.

3.When an exception occurs, JVM will search for matching catch block.

4.If no catch block is found in current method, the method is popped out from stack and exception is passed to caller.

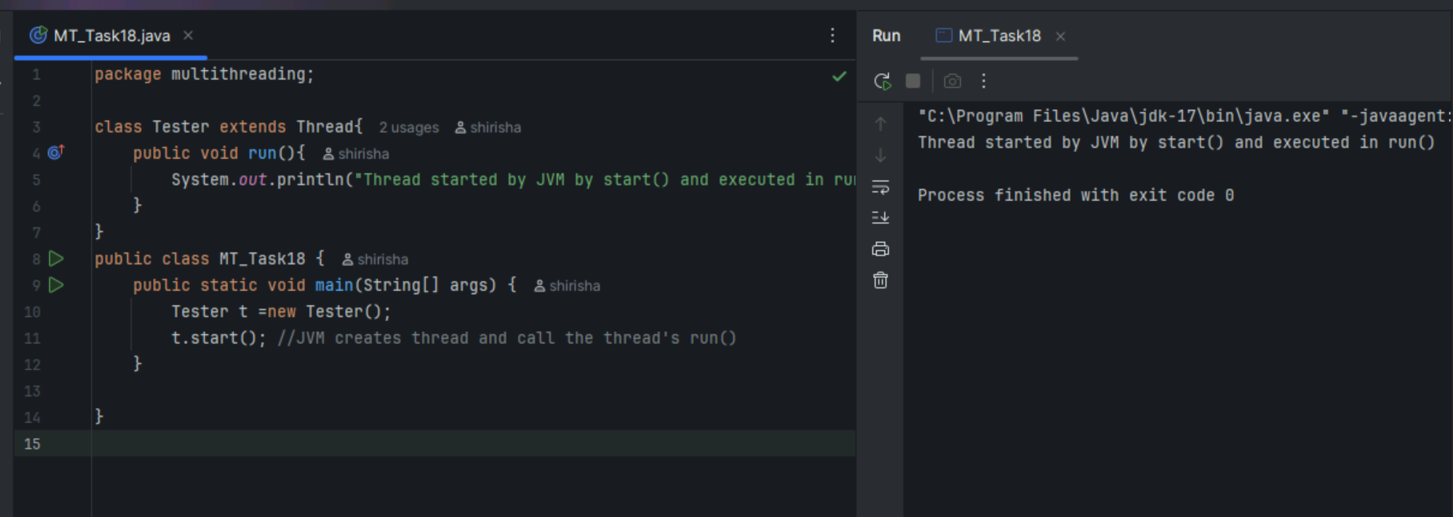
5.This goes on up the call stack method by method, until a handler is found or stack is empty.

6.If the exception is uncaught, JVM prints stack trace showing the sequence of method calls from error back to the start.

So the Stack trace shows each method, class and line number involved, helping developers trace the origin of the error.

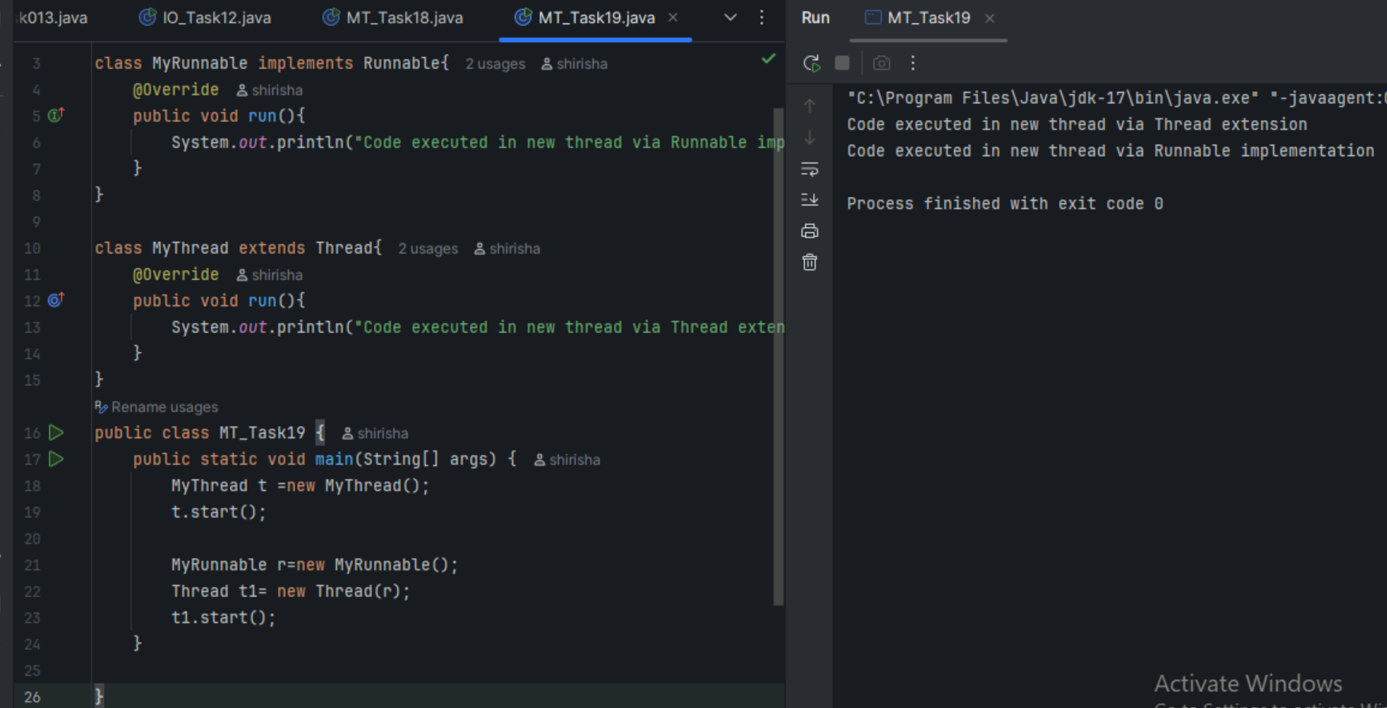
Task 18:

start() and run() difference



Task 19:

Thread execution flow



Task 20

Thread Stack Trace

