MultiThreading

Multitasking

Multitasking allows several activities to occur concurrently on the computer.

* process-based multitasking
* Thread-based multitasking

Process based multitasking: Allows processes (i,e programs) to run concurrently on the computer.

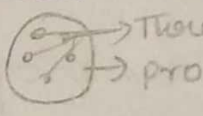
Eg:- Running the MS paint while also working with the word processor.

Thread based Multitasking: Allows part of the same program to run concurrently on the computer.

Eg:- MS word i,e printing & formatting text at the same time.

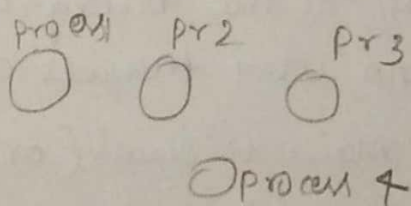
Imp Thread V/S process

Thread → A thread is the smallest unit of execution within a process. Multiple threads share the same memory and run in parallel.



process → A process is an independent program, execute with its own memory space.

Multiple processes run separately and don't share memory.



Why MultiThreading?

* In a single-threaded environment, only one task at a time can be performed.
* CPU cycles are wasted, for example, when waiting for user input.
* Multitasking allows idle CPU time to be put to good use.

⇒ The Main Thread

* When you run a java program, a special thread (called the main thread) is automatically started to run the main() method.
* If your program does not create any extra threads, it will stop running as soon as the main() method finishes.
* If your program creates additional threads, they are called child threads bcoz they are created by the main thread (parent thread)
* User threads: Keep the program running until they finish.
* Daemon Threads: These are background threads that do tasks like garbage collection.

  Run in the background but automatically stop when no user threads are left.
* If the main thread (or any user thread) finishes, daemon threads will also stop immediately, even if they are in the middle of a task.

* setDaemon(boolean) ⇒ used to create the Daemon thread

Thread Creation

2 ways

a) Implementing the java.lang.Runnable Interface

b) Extending the java.lang.Thread class.

Eg:-

```
class Thread1 extends Thread {
    @Override
    public void run() {
        for (int i=0; i<5; i++) {
            S.o.pln ("Inside thread1" + i);
    } } }
public class Day33 {
    public static void main(String[] args) {
        S.o.pln (" Main thread running");
    Thread1 t1 = new Thread1();
    t1.start();
    S.o.pln (" Main thread exiting");
} }
```

The main thread does not wait for t1 to finish.

* The t1 thread starts running, but the main thread continues its own execution and exits before Thread1 finishes

* Threads are executed based on the JVM Schedule

O/P:-

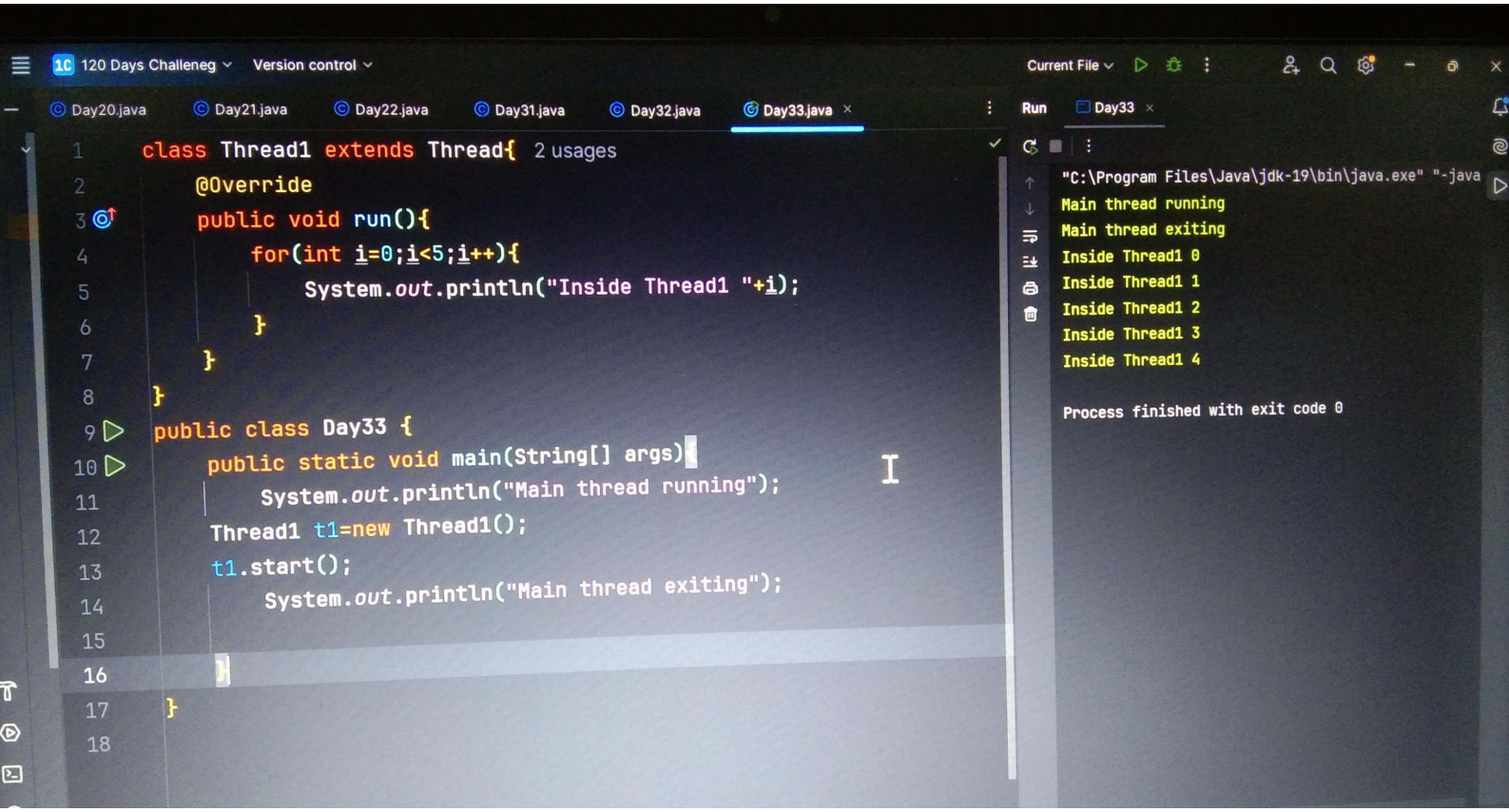
```
Main thread running
Main thread exiting
Inside Thread1 0
Inside Thread1 1
Inside Thread1 2
Inside Thread1 3
Inside Thread1 4
```

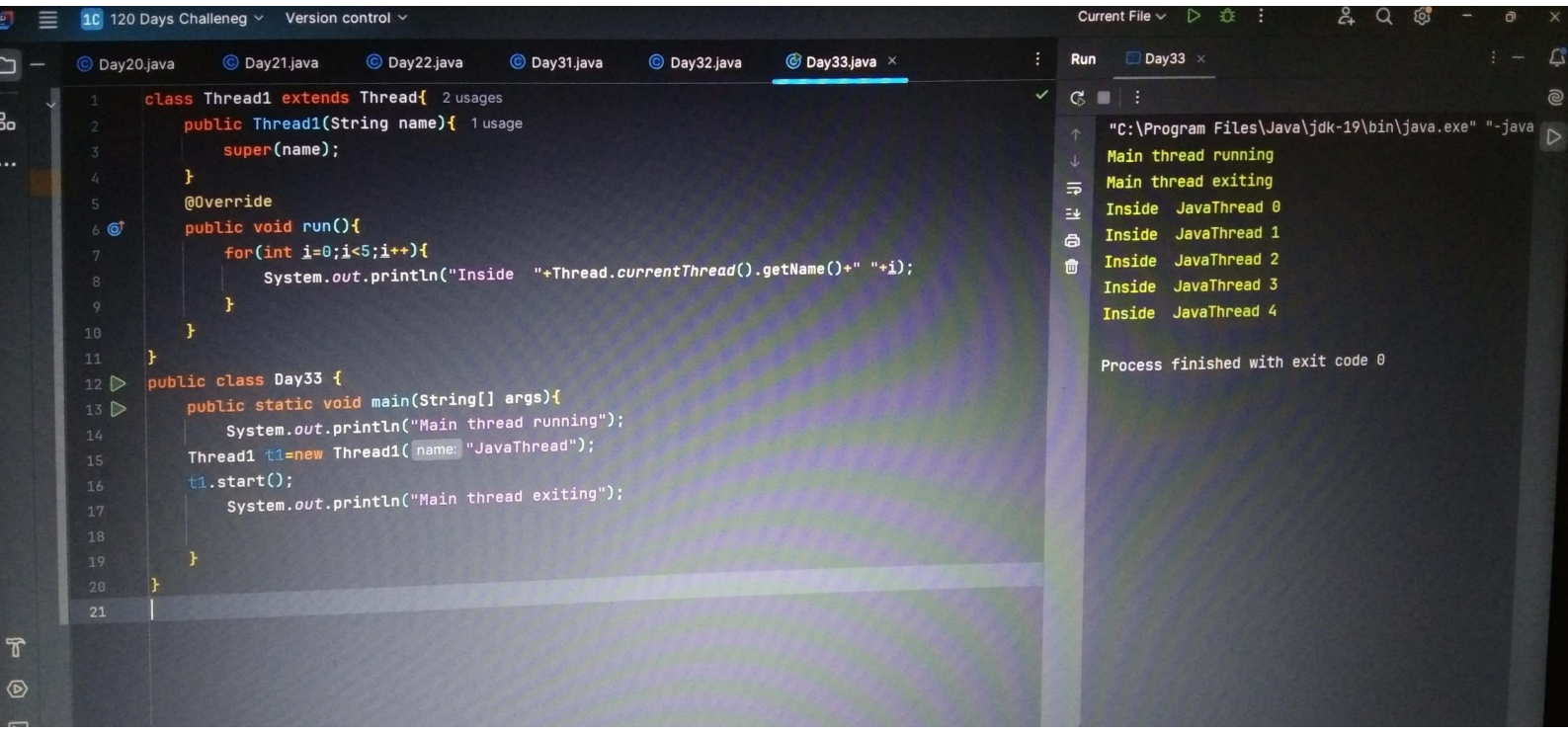
59:-

```
class Thread1 extends Thread {
    public Thread1(String name) {        } → constructor
        super(name);
    }
@Override
public void run() {
    for(int i=0; i<5; i++) {        ↑ Represent the current thread name
        s.o.pln("Inside " + Thread.currentThread().
                                ↓ getName() + " " + i);
    }
}
public class Day33 {
    public static void main(String[] args) {
        System.out.println("Main thread running");
Thread t1 = new Thread1("JavaThread");
t1.start();
s.o.pln("main thread exiting");
}
}
```

If u mention Thread1 it represents thread1 only but if u mention Thread(main) ⇒ represents currently running thread.

O/P:- main thread running
main thread exiting
Inside JavaThread 0
Inside JavaThread 1
Inside JavaThread 2
Inside JavaThread 3
Inside JavaThread 4


120 Days Challeneg
Version control
Current File
Day20.java
Day21.java
Day22.java
Day31.java
Day32.java
Day33.java
Run
Day33
class Thread1 extends Thread{ 2 usages
@Override
public void run(){
for(int i=0;i<5;i++){
System.out.println("Inside Thread1 "+i);
}
}
}
public class Day33 {
public static void main(String[] args)
System.out.println("Main thread running");
Thread1 t1=new Thread1();
t1.start();
System.out.println("Main thread exiting");
}
}
"C:\Program Files\Java\jdk-19\bin\java.exe" "-java
Main thread running
Main thread exiting
Inside Thread1 0
Inside Thread1 1
Inside Thread1 2
Inside Thread1 3
Inside Thread1 4
Process finished with exit code 0


120 Days Challeneg
Version control
Current File
Day20.java
Day21.java
Day22.java
Day31.java
Day32.java
Day33.java
class Thread1 extends Thread{ 2 usages
public Thread1(String name){ 1 usage
super(name);
}
@Override
public void run(){
for(int i=0;i<5;i++){
System.out.println("Inside "+Thread.currentThread().getName()+" "+i);
}
}
}
public class Day33 {
public static void main(String[] args){
System.out.println("Main thread running");
Thread1 t1=new Thread1(name: "JavaThread");
t1.start();
System.out.println("Main thread exiting");
}
}
Run
Day33
"C:\Program Files\Java\jdk-19\bin\java.exe" "-java
Main thread running
Main thread exiting
Inside JavaThread 0
Inside JavaThread 1
Inside JavaThread 2
Inside JavaThread 3
Inside JavaThread 4
Process finished with exit code 0