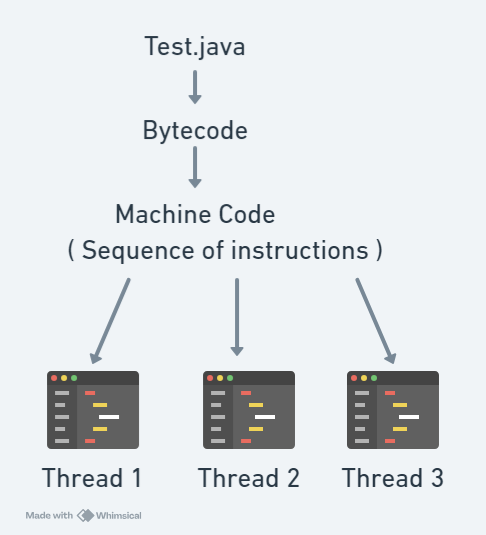


Process

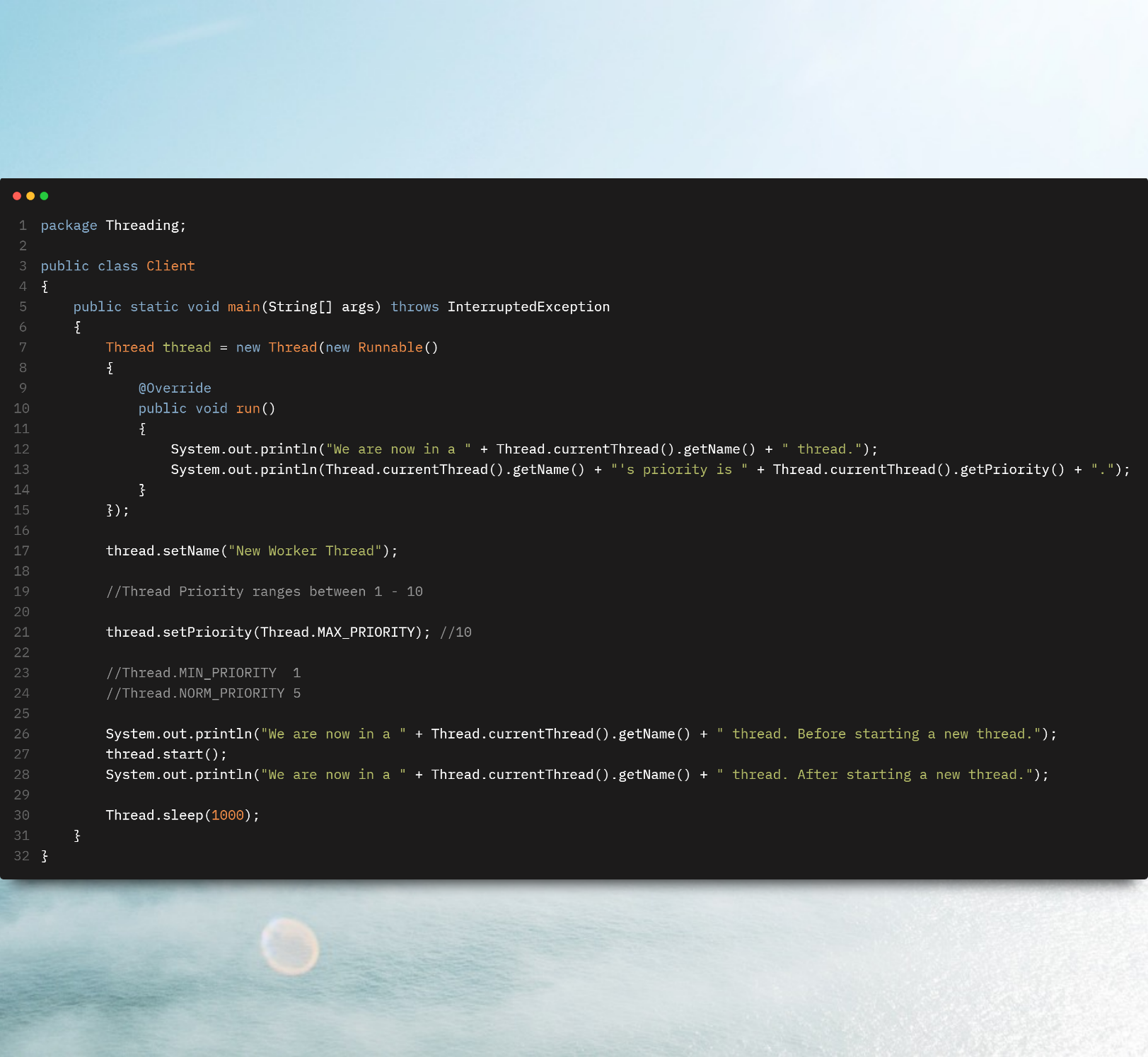
* Process is an instance of a program that is getting executed.
* It has its own resources like memory, thread etc. OS allocates the resources to process when it is created.
* Step1: Compilation javac Test.java. It generates the bytecode that can be executed by JVM.
* Step2: Exection java Test. At this point J V M starts the new process.

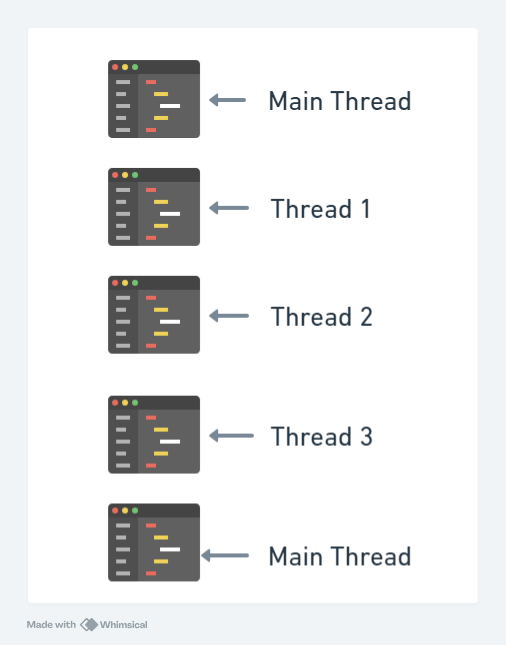
Thread

* Thread is a lightweight process.
* It is the smallest sequence of instructions that are executed by CPU independently.
* A process can have multiple threads. When a process is created it starts with an initial thread called as main thread. From main thread we can create multiple threads to perform task concurrently.



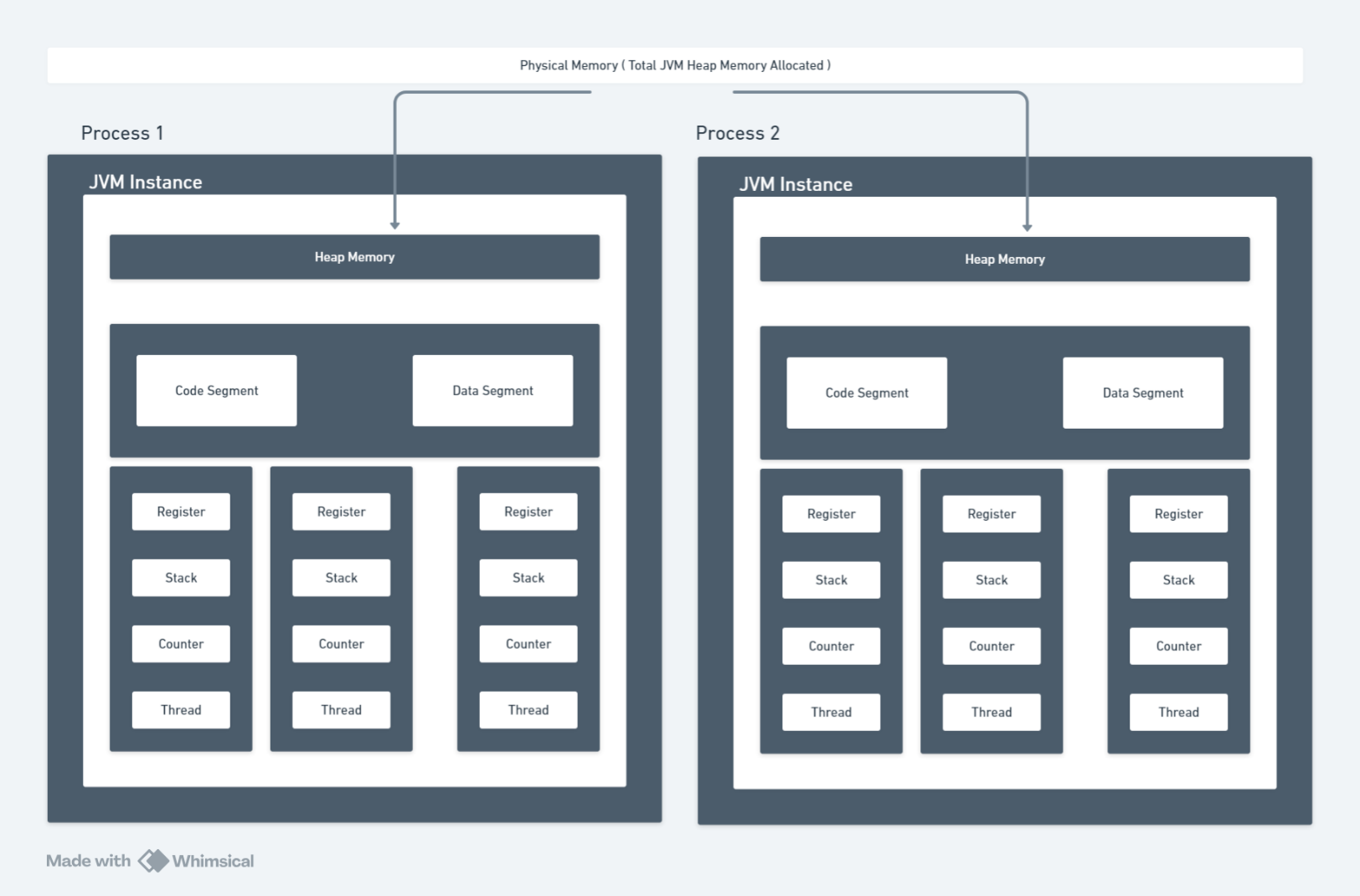
Thread Capabilities



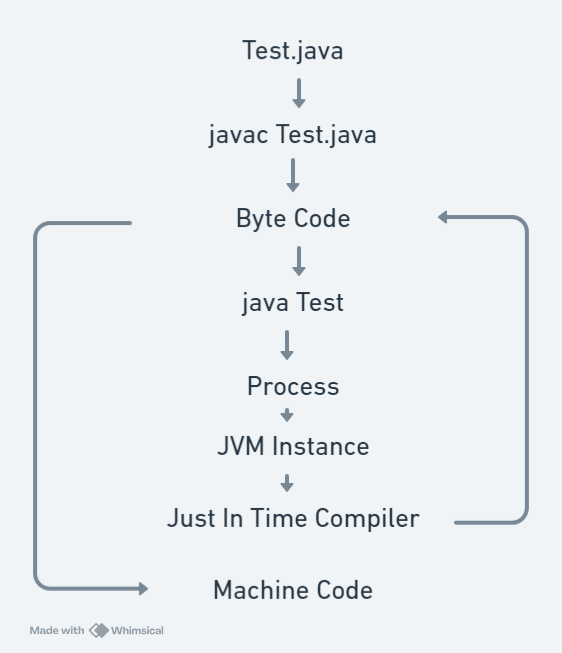


JVM

Whenever a process is created, a new JVM instance is allocated to that process. Because to execute a program, a process needs resources. Therefore, it is the responsibility of JVM to provide all resources to a process.



Heap memory, Code segment and data segment are shared among all the threads while Register, Stack and Counter are local to each thread.



Code Segment

* It stores the compiled Byte Code or Machine Code of the Java Program.
* All threads within the same process share the same code segment.
* It is read only i.e. can not be modified by a thread.

Data Segment

* It stores the global and static variables.
* All threads within the same process share the same data segment.
* Threads can read and modify the same data. Therefore, synchronization is required between multiple threads.

Heap

* Objects created runtime using new keyword are allocated in heap.
* All threads within the same process share the same heap memory but heap memory is not shared among different process.

Stack

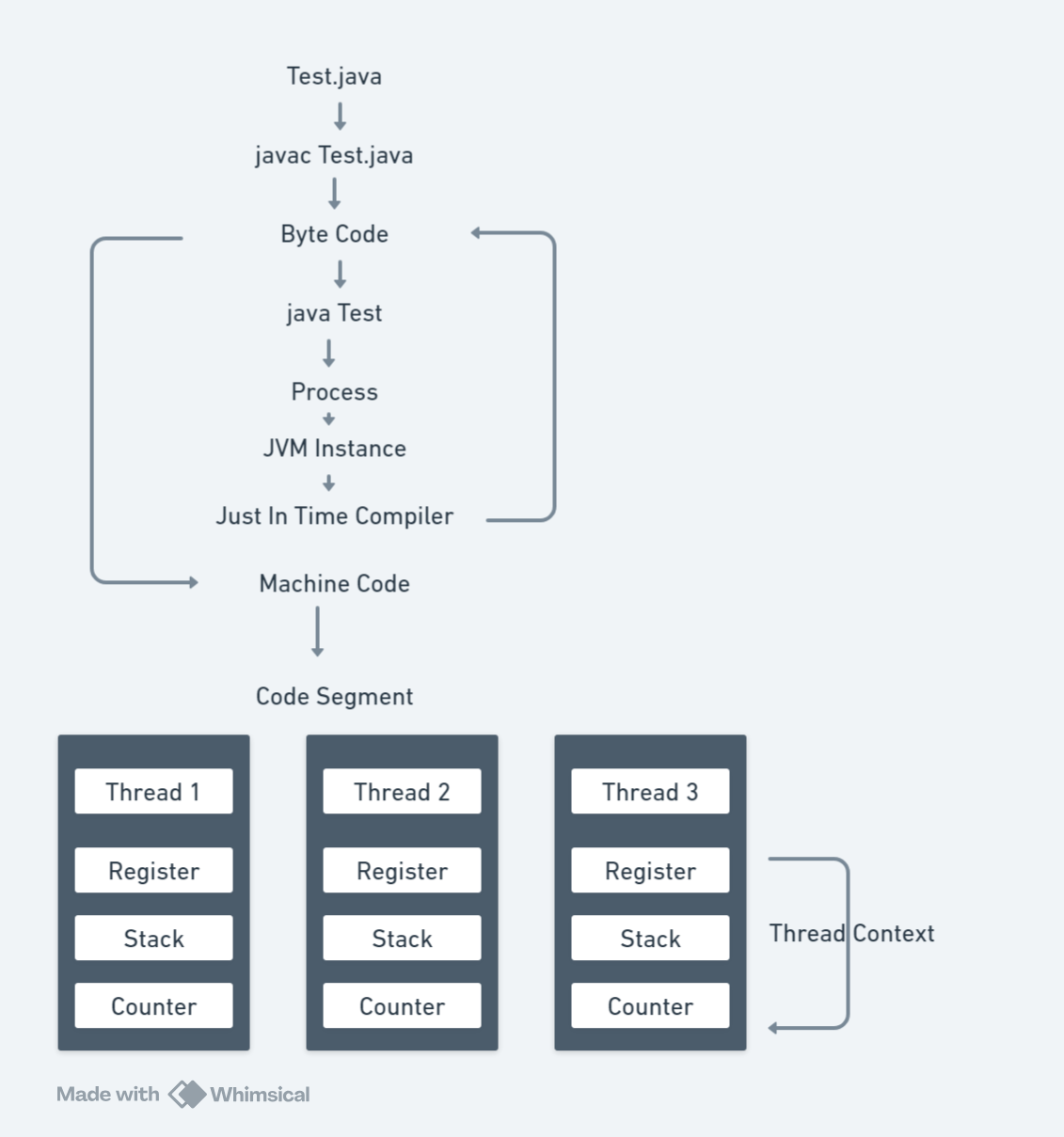
* Each thread has its own stack.
* It manages method calls and local variables.

Register

* Register is used to store intermediate values.
* It also helps in context switching.
* Each thread has its own register.

Program Counter

* It points to the instruction which is getting executed in the code segment.
* It increments its counter after successfully execution of the instruction.



Machine Code gets stored to Code Segment

Thread Context

* Each thread has its own context i.e. register, stack, program counter etc.
* When a thread runs, it uses this context to track the state of execution.

Context Switching

The OS keeps track of the context of each thread. So that it can switch between threads efficiently. This switching process is called Context Switching.

CPU Scheduling

* The OS manages which thread should run on CPU through a process is called as scheduling.
* Sometimes the scheduling task is done by J V M and sometimes it is done by OS.
* The scheduler decides which thread should run at a given time based on variety of algorithms like Round Robin algorithm, Priority based scheduling etc.

Running on CPU

* When a thread is scheduled to run, the CPU loads the thread’s context and other resources needed for execution.
* The CPU executes the thread step by step.

Pre-emption and Time Slices

* The OS allocates a time slice for each thread to run on CPU.
* When this time slice expires, the OS can pre-empt the thread and perform a context switch to another thread.