Threads

* A thread is a single sequential flow of control within a program.
* A single thread can handle only one process at one time multiple thread can access multiple process at same time.

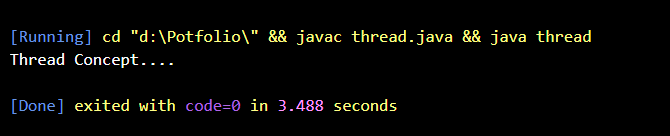
There are two ways to create a thread:

1. By extending Thread class.
2. By implements Runnable interface.

Syntax (Thread Class):



Output:

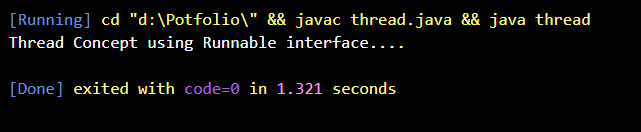


* Run is a abstract method form runnable interface internally thread class can implements the runnable so that is mandatory to implement run method then only we can create separate thread.
* When we call start method then only thread will start and it call run method to perform tasks and we archive multithreading concept.

Syntax (Runnable):



Output:



* When we are creating a thread with Runnable interface first we have to create Object for Thread class then only another thread will be created and we have to pass object ref of class inside thread constructor when only our new thread will be created.

Thread constructors:

* Thread()
* Thread(String name)
* Thread(Runnable r)
* Thread(Runnable r, String name)

Start:

* The **start () method** of Thread class is used to start a newly created thread. It performs the following tasks:
  1. A new thread starts (with new callstack).
  2. The thread moves from New state to the Runnable state.
  3. When the thread gets a chance to execute, its target run() method will run.

Methods In Thread:

1. **public void run():** is used to perform action for a thread.
2. **public void start():** starts the execution of the thread. JVM calls the run() method on the thread.
3. **public void sleep(long miliseconds):** Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.
4. **public void join():** waits for a thread to die.
5. **public void join(long miliseconds):** waits for a thread to die for the specified miliseconds.
6. **public int getPriority():** returns the priority of the thread.
7. **public int setPriority(int priority):** changes the priority of the thread.
8. **public String getName():** returns the name of the thread.
9. **public void setName(String name):** changes the name of the thread.

Synchronized:

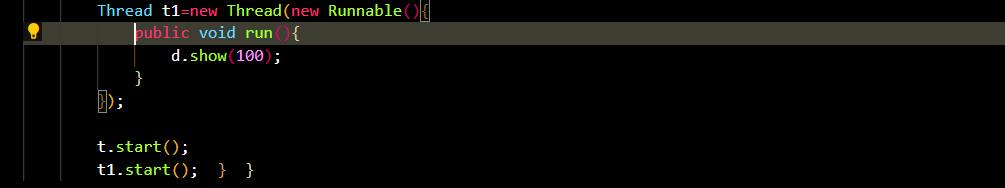
* Synchronization in Java is the capability to control the access of multiple threads to any shared resource.

Uses:

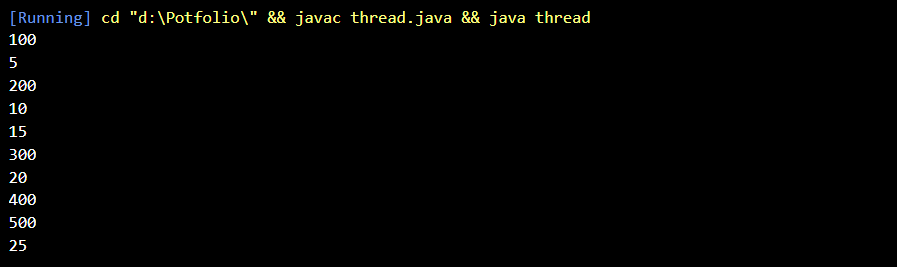
* 1. To prevent thread interference.
  2. To prevent consistency problem.

Without Synchronized:

Example:

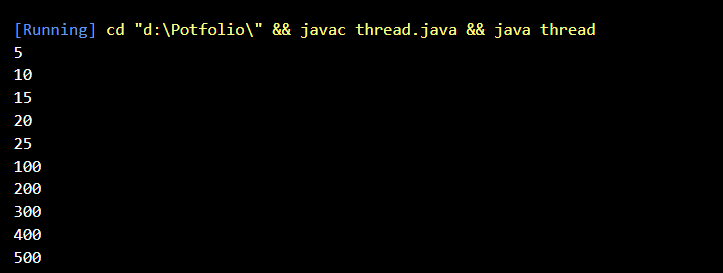
Output:



With Synchronized:



Output:



Inter-thread Communication:

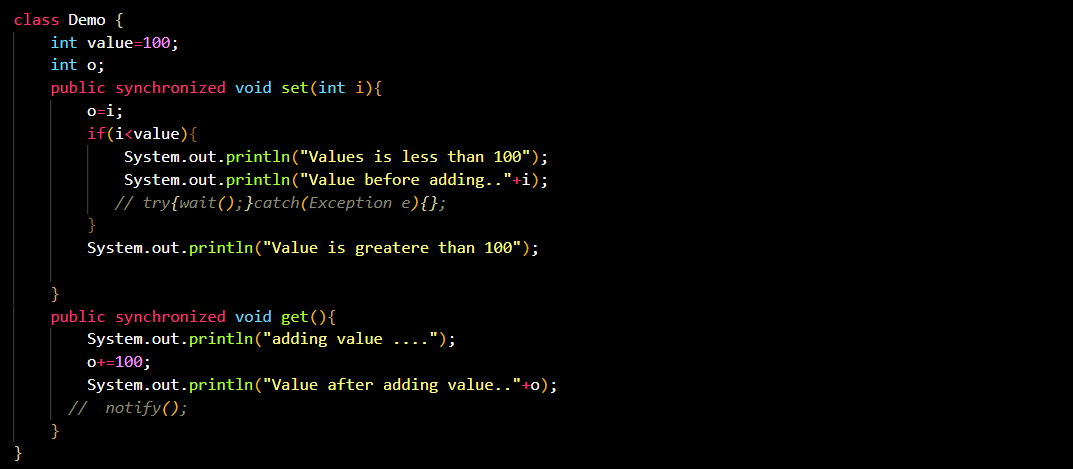
* Inter thread communication is allow synchronized threads to communicate each other.
* Methods to achieve inter thread communication.

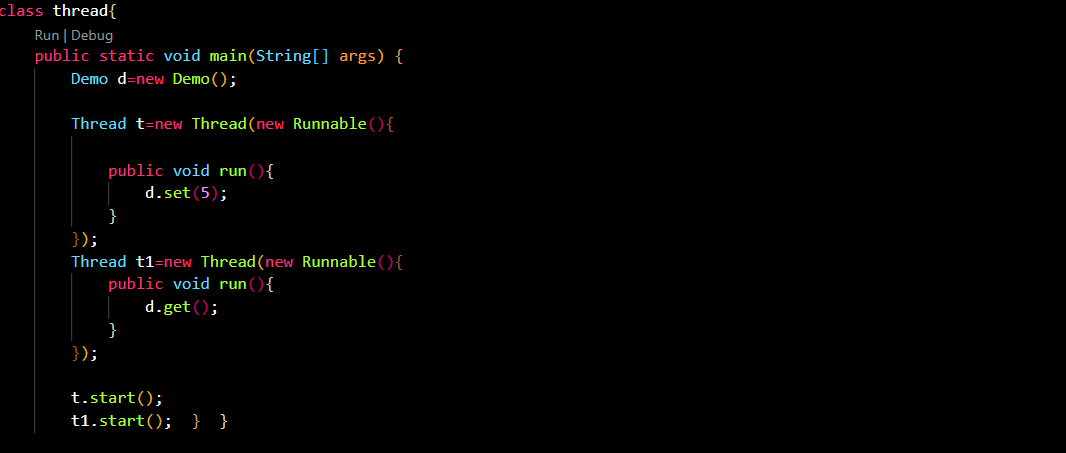
notify

wait

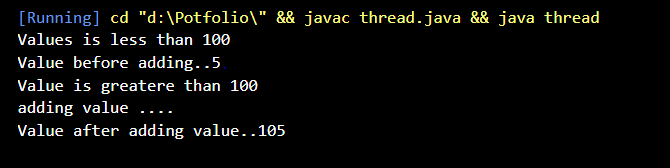
notifyAll

Example:



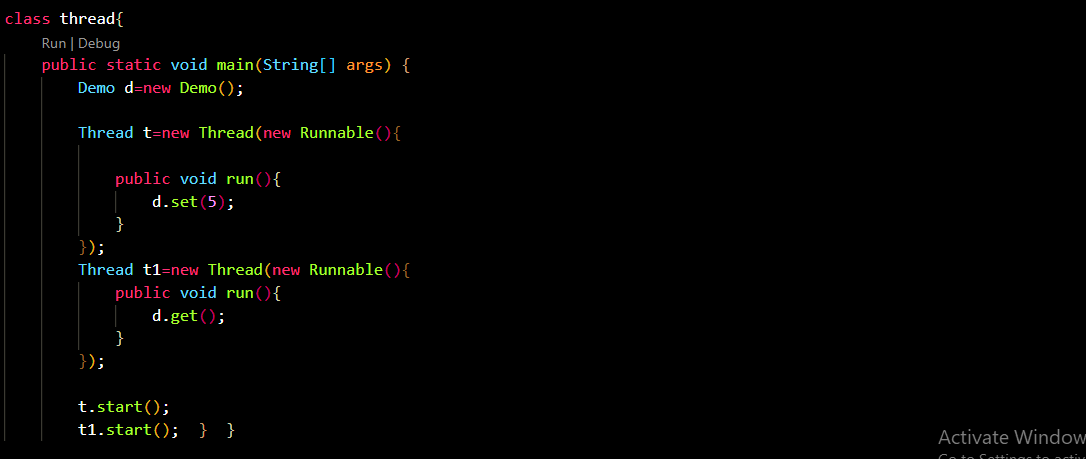
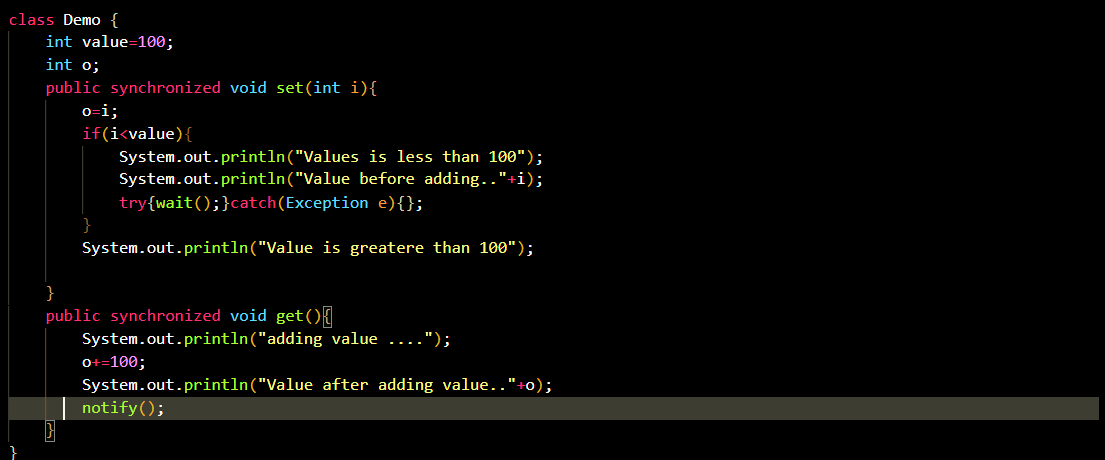


Output:



* In above example we are using two threads on two synchronized methods second thread is wait till first thread completes its task so it printing all outputs in first method.

Example (With wait () and notify ()):



Output:

