*public class* FixedThreadPoolExample {
 *public static void* main(String[] args) {
 *// Create a FixedThreadPool with 5 threads
 ExecutorService* executor = Executors.*newFixedThreadPool*(5);

 *// Submit tasks to the executor
 for* (*int* i = 0; i < 10; i++) {
 *final int* taskNumber = i;
 executor.submit(() -> {
 System.*out*.println("Task " + taskNumber + " executed by thread: " + Thread.*currentThread*().getName());
 });
 }

 *// Shutdown the executor* executor.shutdown();
 }
}

Task 1 executed by thread: pool-1-thread-2

Task 2 executed by thread: pool-1-thread-3

Task 0 executed by thread: pool-1-thread-1

Task 3 executed by thread: pool-1-thread-4

Task 4 executed by thread: pool-1-thread-5

Task 8 executed by thread: pool-1-thread-3

*public class* CachedThreadPoolExample {
 *public static void* main(String[] args) *throws* InterruptedException {
 *// Create a CachedThreadPool
 ExecutorService* executor = Executors.*newCachedThreadPool*();

 *// Submit tasks to the executor
 for* (*int* i = 0; i < 100; i++) {
 executor.submit(() -> {
 System.*out*.println("Task executed by thread: " + " -> " + Thread.*currentThread*().getName());
 });
 }

 *// Shutdown the executor service when tasks are completed* executor.shutdown();
 }
}

Task executed by thread: -> pool-1-thread-1

Task executed by thread: -> pool-1-thread-5

Task executed by thread: -> pool-1-thread-2

Task executed by thread: -> pool-1-thread-4

Task executed by thread: -> pool-1-thread-3

*public class* ScheduledThreadPoolExample {
 *public static void* main(String[] args) *throws* InterruptedException {
 *// Create a ScheduledThreadPoolExecutor with 5 core threads
 try* (*ScheduledExecutorService* scheduledThreadPool = Executors.*newScheduledThreadPool*(5)) {

 *// Schedule a task to run repeatedly every 2 seconds, starting after an initial delay of 3 seconds* scheduledThreadPool.scheduleAtFixedRate(() -> {
 System.*out*.println("Task executed every 2 seconds");
 }, 3, 2, TimeUnit.*SECONDS*);

*// Schedule a task to run after a delay of 1 second* scheduledThreadPool.schedule(() -> {
 System.*out*.println("Task executed after 1 second");
 }, 1, TimeUnit.*SECONDS*);

*// Thread.sleep(10000);
// scheduledThreadPool.shutdown();
 try* {
 scheduledThreadPool.awaitTermination(5, TimeUnit.*SECONDS*);
 *// Shutdown the executor after 10 seconds* scheduledThreadPool.shutdown();
 } *catch* (InterruptedException e) {

 e.printStackTrace();
 }
 }
 }
}

Task executed after 1 second

Task executed every 2 seconds

Task executed every 2 seconds

*public class* SemaphoreExample {
 *private static final int THREAD\_COUNT* = 5;

 *public static void* main(String[] args) {
 *// Create a Semaphore with permits for THREAD\_COUNT concurrent threads* Semaphore semaphore = *new* Semaphore(*THREAD\_COUNT*);

 *// Create and start multiple threads
 for* (*int* i = 0; i < *THREAD\_COUNT* \* 2; i++) {
 Thread thread = *new* Thread(*new* Worker(semaphore));
 thread.start();
 }
 }

 *static class* Worker *implements Runnable* {
 *private final* Semaphore semaphore;

 Worker(Semaphore semaphore) {
 *this*.semaphore = semaphore;
 }

 @Override
 *public void* run() {
 *try* {
 *// Acquire a permit from the semaphore* semaphore.acquire();
 System.*out*.println(Thread.*currentThread*().getName() + " has acquired a permit.");

 *// Simulate some work* Thread.*sleep*(1000);

 *// Release the permit back to the semaphore* semaphore.release();
 System.*out*.println(Thread.*currentThread*().getName() + " has released the permit.");
 } *catch* (InterruptedException e) {
 e.printStackTrace();
 }
 }
 }
}

Thread-2 has acquired a permit.

Thread-3 has acquired a permit.

Thread-1 has acquired a permit.

Thread-5 has acquired a permit.

Thread-0 has released the permit.

Thread-6 has acquired a permit.

Thread-1 has released the permit.

Thread-4 has released the permit.