##### **Thread**

A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.

Thread class represent a vitual thread.

One thread object can only be ****started once****.

public class MyThread extends Thread {

public void run() {

System.out.println("Hello World");

}}

MyThread t = new MyThread();

t.start(); // start the thread to print

//////////////////////Second Way - using Runnable///////////public class MyThread2 implements Runnable {

public void run() {

System.out.println("Hi");

}}

Thread t = new Thread(new MyThread2());

t.start(); // start

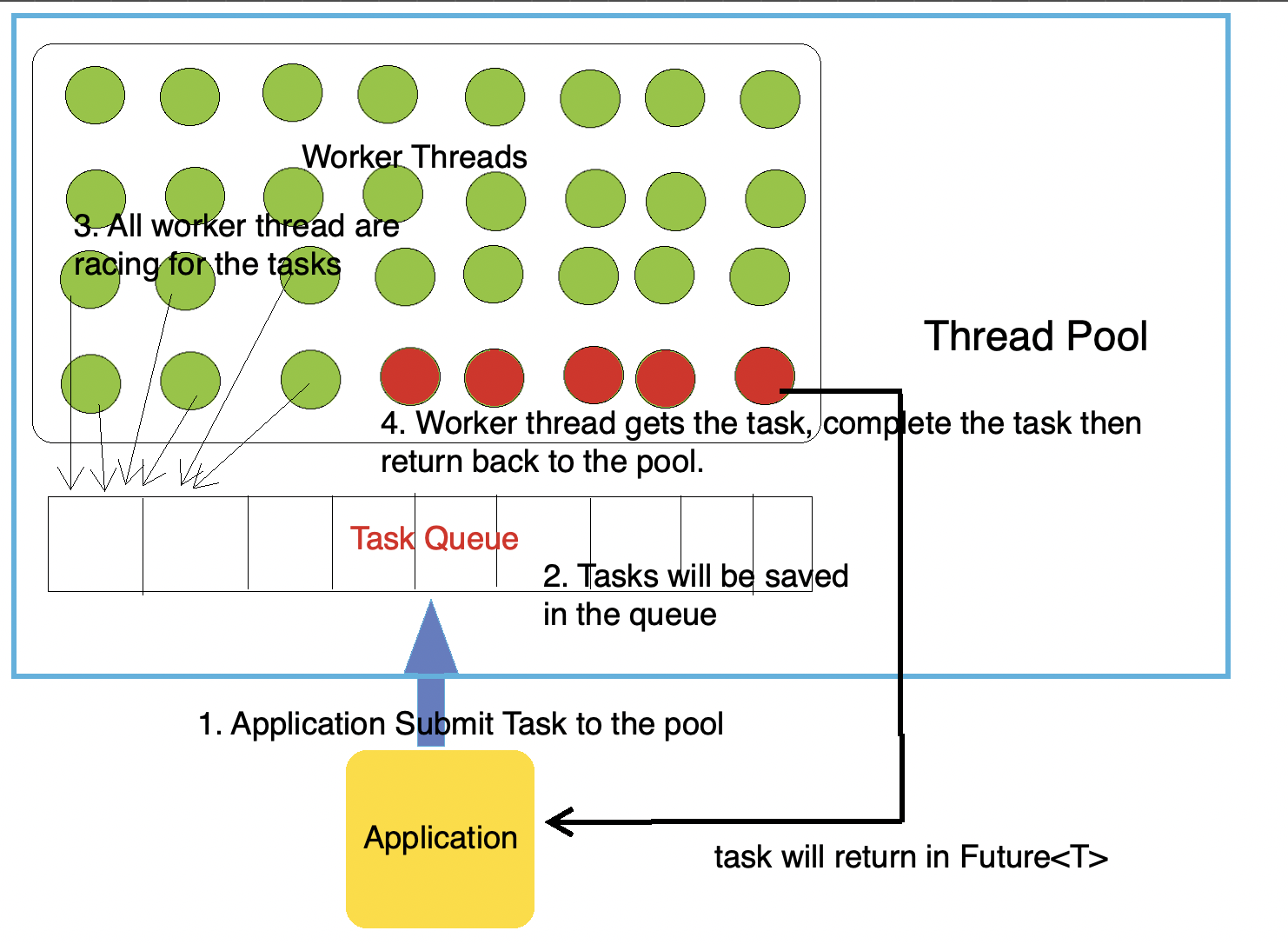
run() method vs start() method

run() method executes in the current thread.

start() method which is pre-implemented executes the run() method in a new thread. so always call start() if you start a thread.

##### **Thread Pool**

Threads are objects. It takes memory and consumes computation power. To reduce the cost of creating and destroying new threads, and also to confine the memory usage. On the server side applications, we use thread pool to pre-populate certiain number of threads to be reused by different tasks.



class MyCallableClass implements Callable<String> { // means it will return String

public String call() { // can throw exception here

// do a lot of work and get a result = "Hi"

//...

return "Hi";

}}

ExecutorService es = Executors.newFixedThreadPool(30);Future<String> future = es.submit(new MyCallableClass());String result = future.get();

<https://www.youtube.com/watch?v=P7l50T4VypQ>

##### **Thread Safe**

Thread running on its own sequencial order won't know what's going on in other threads. A racing condition will arise logic error and unpredictable results. Thread safety issues usually cannot be detected in compilation time. And sometimes hard to reproduc.

Common ways to make a code thread-safe: 1. Synchronization using lock. 2. Don't change the shared data(Stateless Class).

##### **Synchronzied keyword**

A lock machanism to make code to be executed just in one thread at the same time.

The keyword can be put on method or blocks. When put on blocks, explicit lock should be decleared.

class A {

// put on instance method.

// instance method belong to object.

// For the same object if two threads calling this method.

// they have to run one by one.

// so "this"-current object, is the lock.

public synchronized void doSomething(){

System.out.println("hi");

}

}

A a = new A();

Thread1 call a.doSomething();Thread2 call a.doSomething(); <- they have to run one by one.

A a1 = new A();Thread 1 call a.doSomething();Thread 2 call a1.doSomething(); <- they don't have to wait. because "this" of a and a1 are different.

Volatile Keyword

Concurrent Package

ThreadLocal

<https://www.youtube.com/watch?v=uF_jaZJ-leQ>