Javaプログラマ育成コース
第八回　スレッドとスレッドセーフ

ソフトシンク株式会社
代表取締役
周　順彩
zhousc@soft-think.com

目次

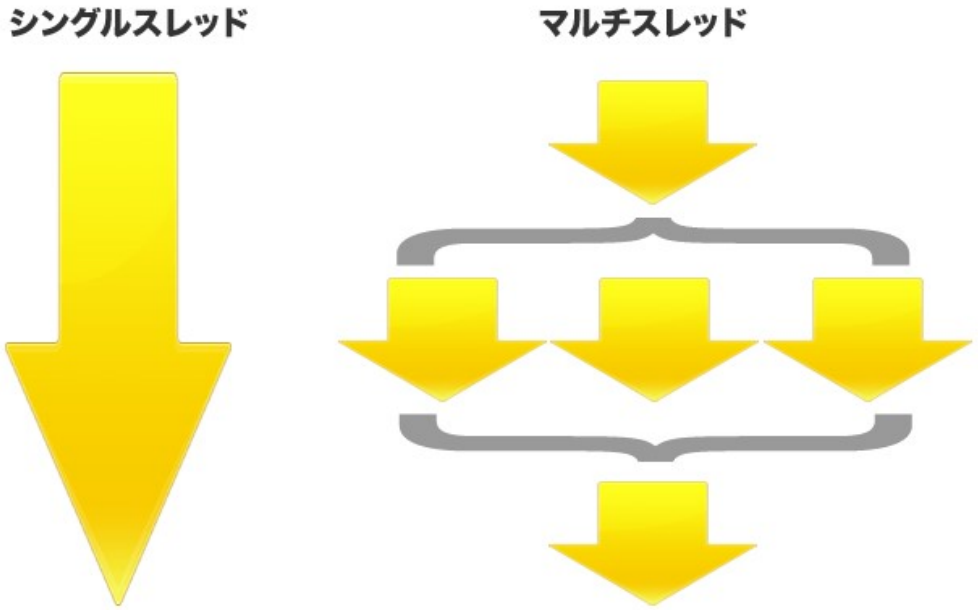
- スレッドの概念
- スレッドの実装
- スレッドセーフ
- 練習課題
- 次回の予習タスク

スレッド

◆ スレッドとは

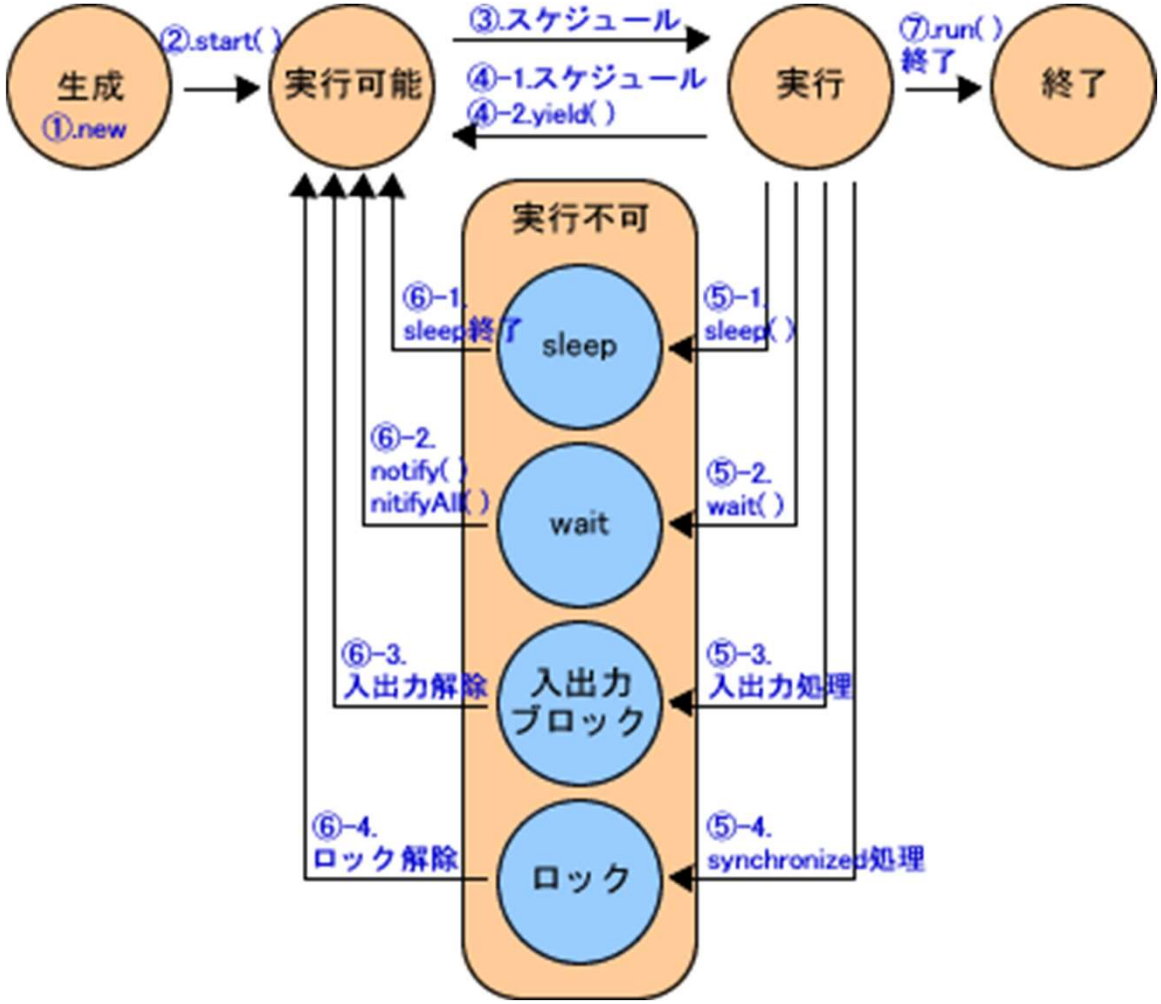
処理を実行する流れの単位をスレッドと呼ぶ。

◆ シングルスレッドとマルチスレッド



スレッドのステータス

スレッドはJVMのスケジューラによって制御され、「実行中」、「実行可能状態」、「実行不可能状態」、「終了状態」といった状態になります。



Objectクラス

Objectクラス中にスレッドに関するメソッドは三つがある。

wait()

wait()を呼び出すと現在のスレッドが待機する。

notify()

notify()を呼び出すとそのオブジェクトのウェイトセットにあるスレッドが1つ再開する。

notifyAll()

notifyAll()を呼び出すと、そのオブジェクトのウェイトセットにある全てのスレッドが再開する。

スレッドの作成

スレッドの作成は下記の二つの方法があります。

- Threadクラスを継承する方法
- Runnableインタフェースを実装する方法

区別:

1. 多重継承
2. 資源共有

メソッド

メソッド名	説明
sleep()	現在実行中のスレッドを、指定されたミリ秒数の間、スリープ(一時的に実行を停止)させる。
get/setPriority()	優先順位を取得/設定する
yield()	現在のスレッドが現在のプロセッサ使用量を譲る用意があることを示す、スケジューラへのヒントです。
join()	このスレッドが終了するのを待機する。

スレッドプール

複数のスレッドをあらかじめ作成して待機させておき、タスクが来たら待っているスレッドにタスクを割り当てて処理を開始させる仕組みをスレッドプールと言います。処理すべきタスクが到着してからモタモタとスレッドを起動するのではなく、スレッドをあらかじめ起動しておき、タスクが割り当てられたらすぐに処理を開始出来るようにする、ということです。

分類:

- CachedThreadPool
- FixedThreadPool
- SingleThreadPool
- ScheduledThreadPool

戻り値

Runnableインターフェイスを実装しスレッドを実行するのみでしたが、
今度はCallableインターフェイスを実装しスレッドから戻り値を受け取れる。

使用するクラス

- Executors
- ExecutorService
- Callable
- Future

デーモンスレッド

◆ デーモンスレッドとは

　　デーモンスレッドはプログラムが終了する際に停止を待たず、全てのユーザースレッドが終了すると自動的に終了する性質を持っていること

◆ 使い方法:

　　setDaemon(true);

◆ 利点:

　　終了処理を意識する必要がない。

◆ 使用上の注意:

　　スレッド内で、後処理が必要な処理(リソース破棄 (データベース接続、一時ファイルなど) を行う場合、使用を控える 。

例外

マルチスレッド環境下で，あるスレッドからスローした例外を別のスレッドでキャッチしたい場合，java.lang.Thread.UncaughtExceptionHandlerインタフェースを実装したクラスを用意し，このインスタンスを例外がスローされるスレッドにセットするということをします。

手順:

1. UncaughtExceptionHandlerインタフェースを実装したクラスを用意する。
2. thread.setUncaughtExceptionHandlerメソッドを呼び出す。
3. 例外発生時の処理

スレッドセーフ

スレッドセーフを考えるときに一番問題なのは、オブジェクトのステート(フィールド)へのアクセス。複数のスレッドによる無秩序なアクセスがあっても、正しい状態を維持し続けられるようにクラスを作ることがすなわち、スレッドセーフなクラスを作成するこになる。

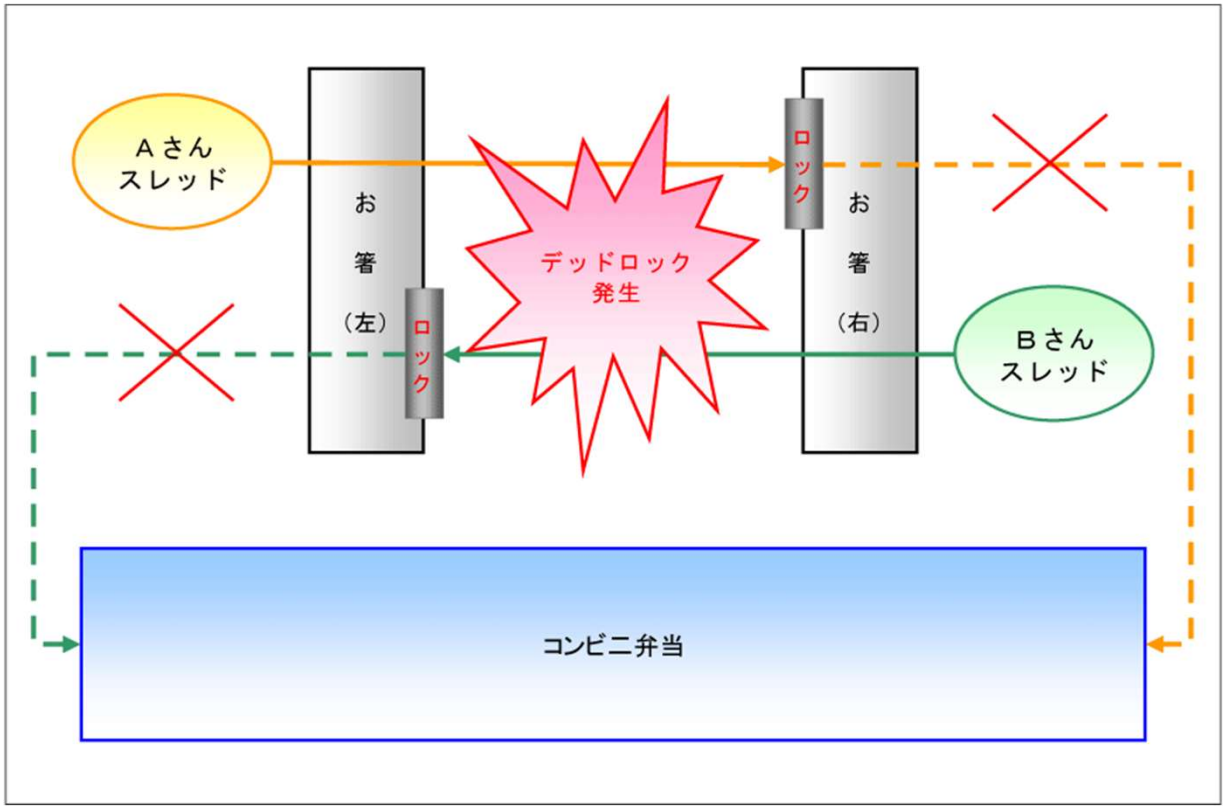
排他制御(ロック)：　複数のプロセス（またはスレッド）が同時に入ることを防ぐことである。

排他制御に関するクラス

1. **synchronized**
2. **ReentrantLock**
3. **AtomicInteger、AtomicLong、AtomicReference**

デッドロック

デッドロックとは、2つのスレッドが2つのロックを取り合い、互いに相手のスレッドがロックを解放するのを待つ状態のことを言います。



デッドロック対策

きちんと設計よりロックは同一順序で要求および解放する。

練習課題

マルチスレッドを利用して、逆ポーランドの実装を改善する。

次回の予習タスク

- ウェブ開発とServlet