

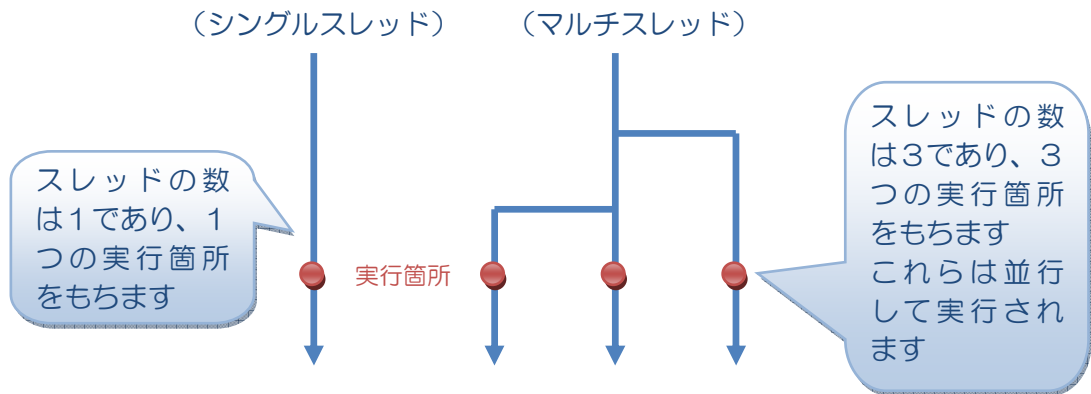
スレッド

スレッドとは

1つの実行箇所をもつ一連の処理の流れです

- ・シングルスレッド → 実行箇所は常に一つです
- ・マルチスレッド → 複数の実行箇所をもちます

Javaではマルチスレッド処理を記述できます



スレッドの生成

スレッドの生成には2通りの方法があります

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

A. Threadクラスを継承する方法

1. Threadクラスを継承してサブクラスを宣言
2. 継承したrun()メソッドをオーバーライド

```
class サブクラス名 extends Thread{
    :
    public void run(){
        別スレッドで実行する処理
    }
    :
}
```

3. mainメソッド内で別スレッドを実行

```
// サブクラスのオブジェクトを生成
サブクラス名 c = new サブクラス名();
// Threadクラスから継承しているstart()メソッドを実行
c.start();
```

Threadクラスの主なメソッド

- public void run(){...}
 - スレッドの処理の記述用
 - オーバーライドして使用
- public void start(){...}
 - 別スレッドでrun()を実行
- public static void sleep(long ms){...}
 - このスレッドを一時停止
- public final void join(){...}
 - このスレッドの終了をまつ

B. Runnable インタフェースを実装する方法

1. Runnable インタフェースを実装してクラスを宣言
2. 実装した抽象メソッド run() をオーバーライド

```
class クラス名 implements Runnable{  
    :  
    public void run(){  
        別スレッドで実行する処理  
    }  
    :  
}
```

Runnable インタフェースのメソッド

```
public abstract void run();  
- スレッドの処理の記述用
```

3. main メソッド内で別スレッドを実行

```
// クラスのオブジェクトを生成  
クラス名 c = new クラス名();  
// このオブジェクトをコンストラクタに渡して Thread クラスのオブジェクトを生成  
Thread th = new Thread(c);  
// Thread クラスの start() メソッドを実行  
th.start();
```

同期

同期とは

複数のスレッドの処理を互いに排他的に行うことです

synchronized メソッド

メソッドの処理は排他的に実行されます

宣言

メソッドの修飾子に synchronized を付加します

※1つのスレッドがあるオブジェクトの synchronized メソッドを実行したら、その処理が終わるまでその他のスレッドはそのオブジェクトのどの synchronized メソッドの実行もできず待たされます

