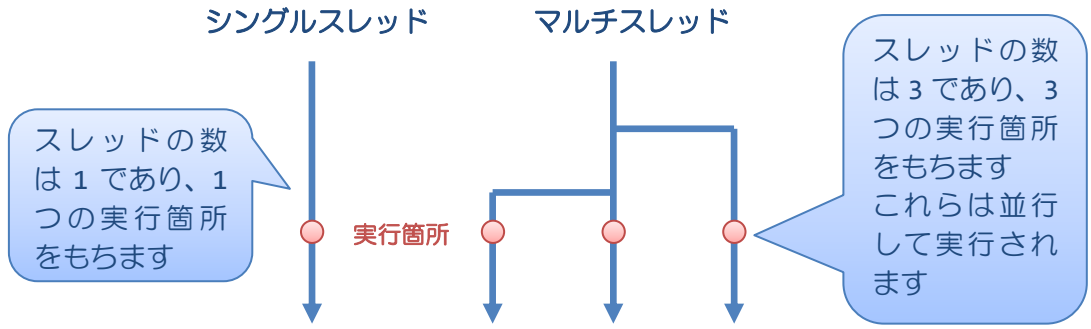
Ｊａｖａプログラミング Ⅱ

１１回目　スレッド

§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 **1 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



■スレッドの起動　新しいスレッドを生成し、**処理の流れを増やすこと**です

§スレッドの起動の方法

スレッドの起動には 2 通りの方法があります
A. Thread クラスを拡張する方法
B. Runnable インタフェースを実装する方法

□ A. Thread クラスを拡張する方法

1. Thread クラスを拡張したサブクラスを宣言
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(){
        新しく起動したスレッドで実行する処理
    }
        :
}
```

3. main()メソッド内で新しいスレッドを起動

```
// 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 クラスの主なメンバ

public void run(){…}
- 新しいスレッドで実行する処理を記述するための空メソッド※1です
- オーバーライドして使用します

※1 Runnable インタフェースを実装したオブジェクトを用いて構築した Thread オブジェクトの場合のために、中には Runnable インタフェースの run()メソッドを呼び出すコードが書かれています。これ以外は何もしません。

public void start(){…}
- 新しいスレッドで run()を実行します

public static void sleep(long m){…}
- このスレッドを m ミリ秒停止します

public final void join(){…}
- このスレッドの終了を待ちます

☐ 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たクラスを宣言
2. 継承した抽象メソッド run()をオーバーライド

```
class クラス名 implements Runnable{
      :
    public void run(){
        新しく起動したスレッドで実行する処理
    }
      :
}
```

Runnable インタフェースのメンバ

`public abstract void run();`

- 新しいスレッドで実行する処理を記述するための抽象メソッドです

3. main()メソッド内で新しいスレッドを起動

```
// 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　　メソッドの処理を排他的に実行します

■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