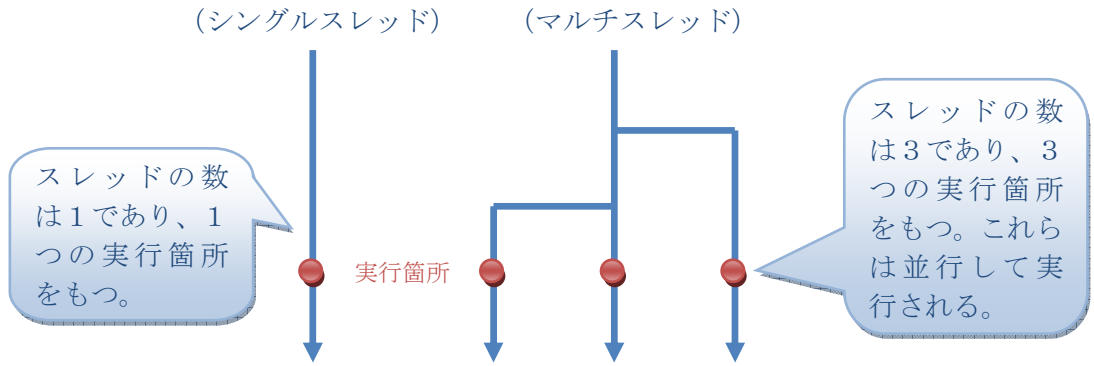
Ｊａｖａプログラミング Ⅱ

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
・シングルスレッド　→ 実行箇所は常に一つ
・マルチスレッド　→ 複数の実行箇所をもつ
Java ではマルチスレッド処理を記述できる



スレッドの生成

スレッドの生成には２通りの方法がある
　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．処理を別スレッドで実行

```
// 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 クラスの主なメンバー

---

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

Runnable ｲﾝﾀﾌｪｰｽのﾒﾝﾊﾞｰ

---

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

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

３．処理を別スレッドで実行

```
// 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

synchronized メソッド　　メソッドの処理は排他的に実行される

宣　　言　　メソッドの修飾子に synchronized を付加

※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る