発展プログラミング

第7回：マルチスレッドプログラミング入門

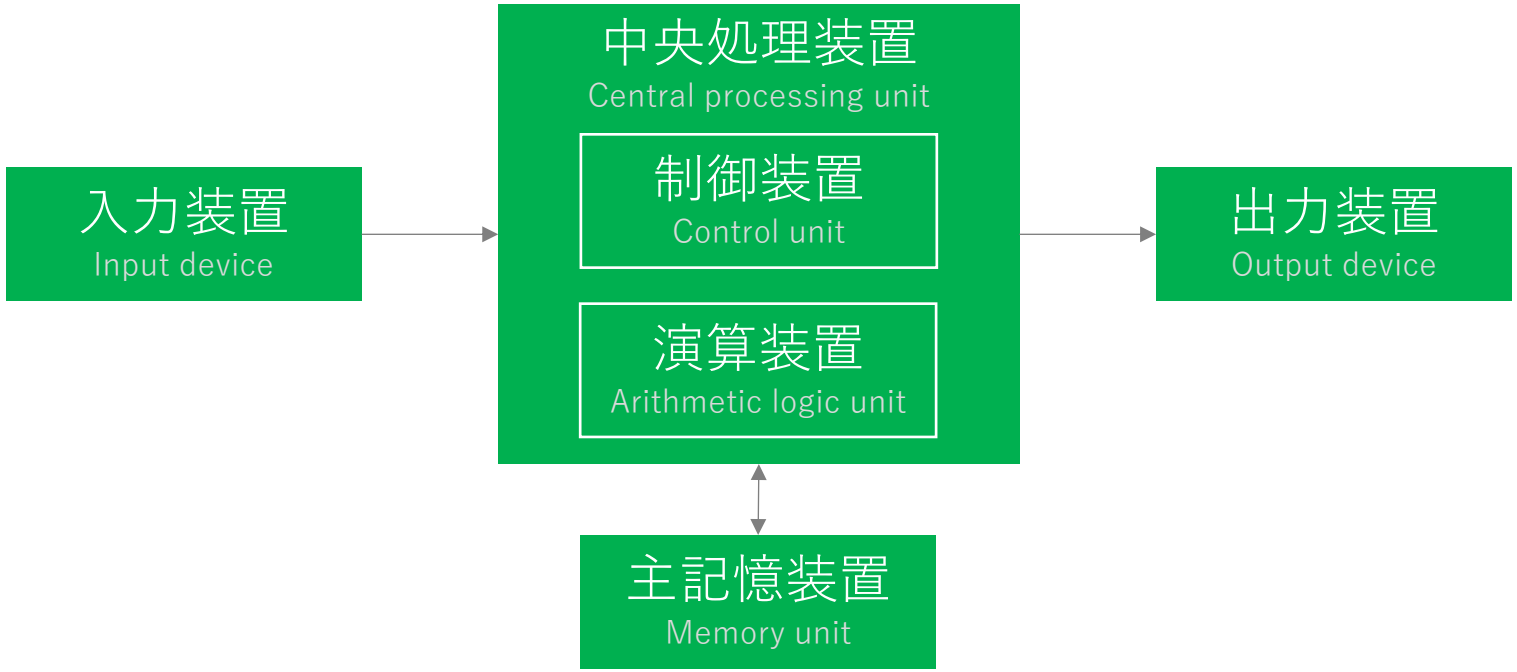
宮田章裕 <miyata.akihiro@nihon-u.ac.jp>

前回講義の復習

計算機アーキテクチャ (1/4)

Computer architecture

- 一般的な計算機を構成する主要装置
  - 制御装置　：演算，記憶装置読み書き，入出力などを制御する
  - 演算装置　：論理演算，四則演算などの演算を行う
  - 主記憶装置：データを記憶する
  - 入力装置　：データを入力する
  - 出力装置　：データを出力する

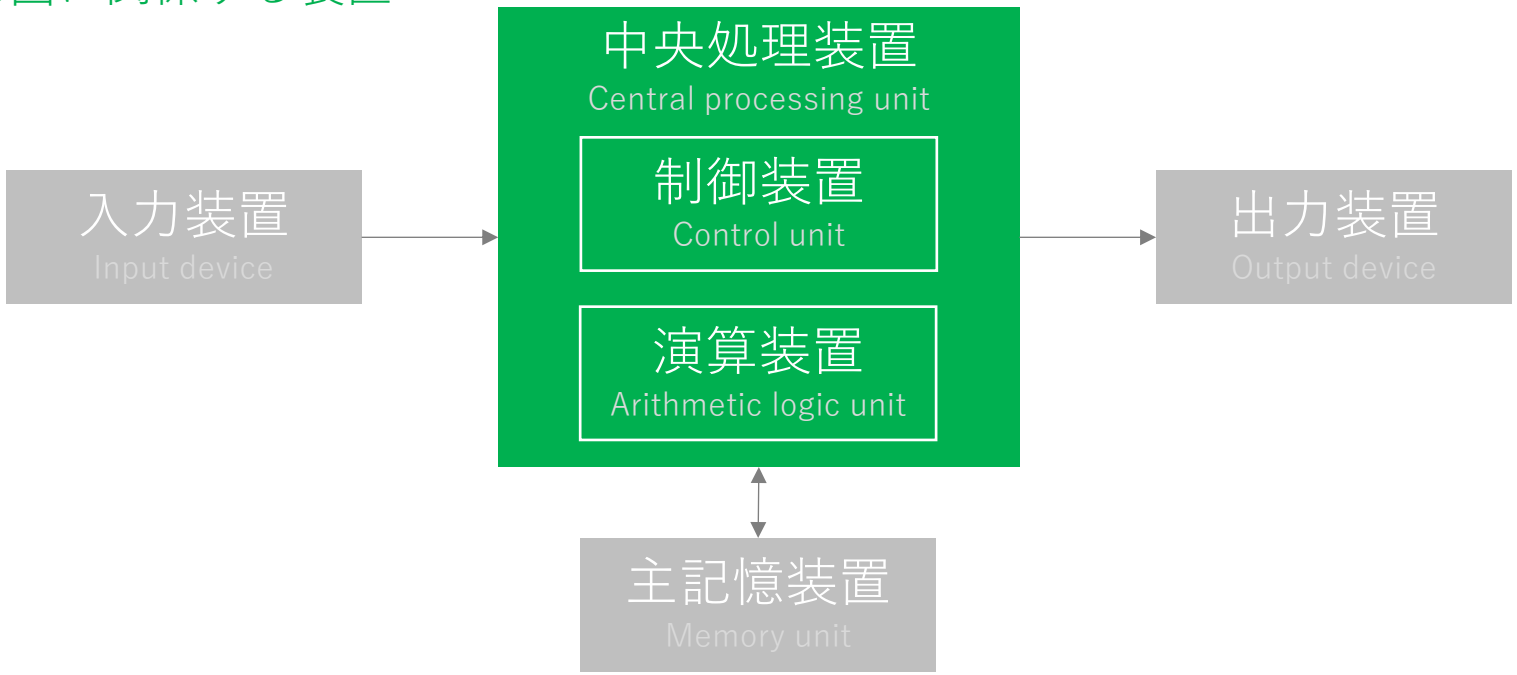


計算機アーキテクチャ (4/4)

Computer architecture

- 一般的な計算機を構成する主要装置
  - 制御装置　：演算，記憶装置読み書き，入出力などを制御する
  - 演算装置　：論理演算，四則演算などの演算を行う
  - 主記憶装置：データを記憶する
  - 入力装置　：データを入力する
  - 出力装置　：データを出力する

本講義第6〜10回に関係する装置



中央処理装置 (1/3)

Central processing unit / CPU

- CPU
  - 制御と演算の役割を担う
  - コア，レジスタ，キャッシュ等からなる

- コア
  - 制御，演算を行うCPUの中核部分
  - 1つのコアで，ある瞬間に同時に行える処理は1つ

    仮想的に1コアで同時に複数処理を行えるHyper-threading technologyなどの技術もあるが，本講義では扱わない。

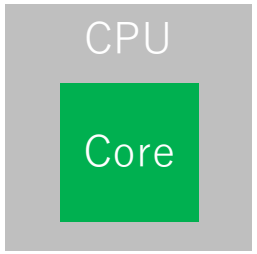
中央処理装置 (2/3)

Central processing unit / CPU

❖シングル／マルチコアCPU

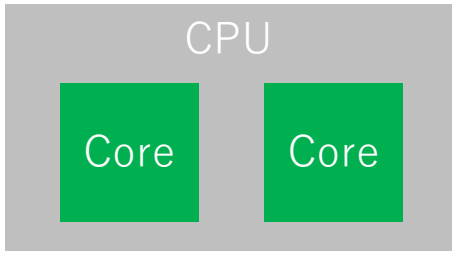
シングルコアCPU

1個のコアを持つCPU



マルチコアCPU

複数個のコアを持つCPU



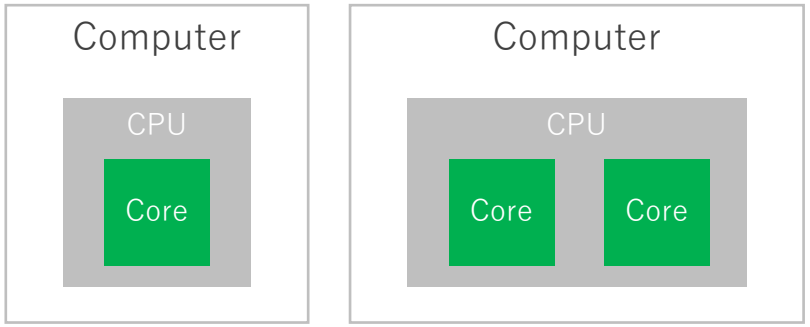
中央処理装置 (3/3)

Central processing unit / CPU

❖シングル／マルチプロセッサ

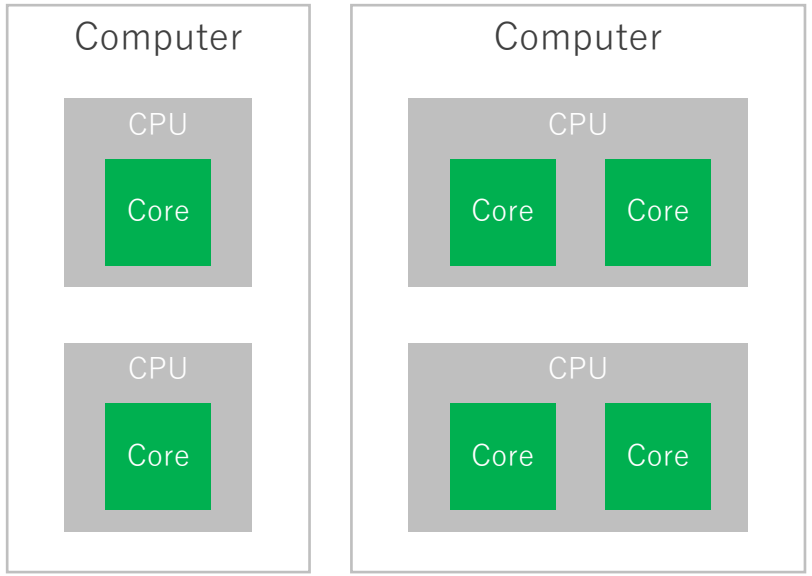
シングルプロセッサ

1個のCPUを搭載した計算機



マルチプロセッサ

複数個のCPUを搭載した計算機



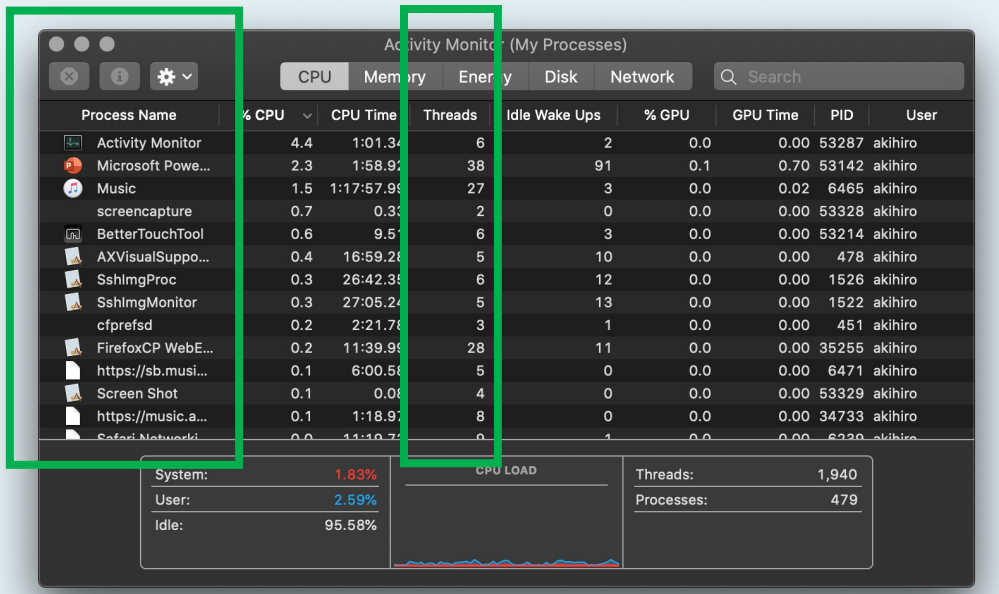
プロセスとスレッド

Process & Thread

- 処理の単位
  - コンピュータ上の各動作はプロセスとして実行される
  - 現代的なOSは，同時に複数のプロセスを実行できる
  - 1つのプロセスは，1つ以上のスレッドから構成される

各プロセスは1つ以上のスレッドから構成される

アプリケーションや
バックグラウンドサービスは
プロセスとして実行される

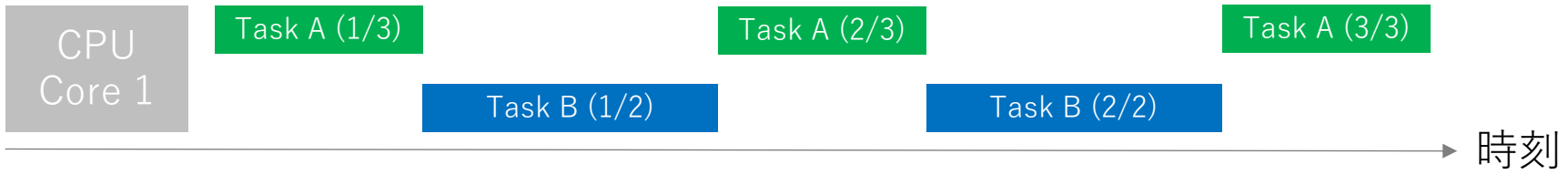


Activity Monitor (macOS)

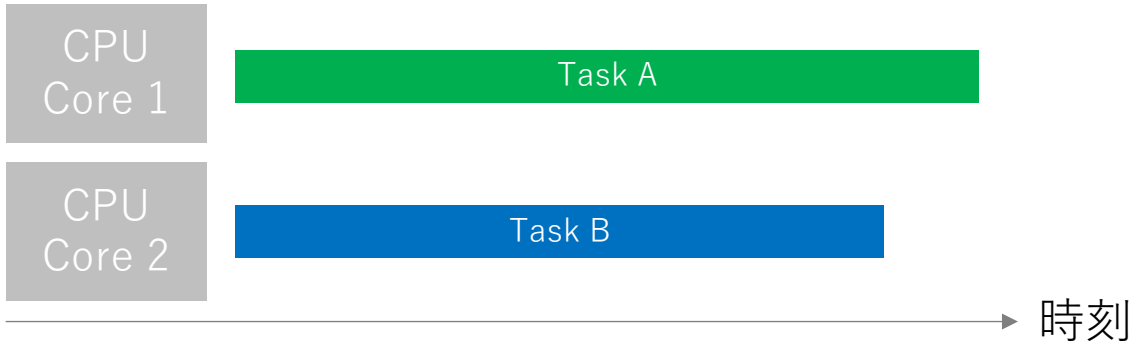
並行と並列

Concurrency vs Parallelism

- 並行 Concurrency
  - 複数の処理を同時に実行状態にする
  - シングルコア／シングルプロセッサでも可能
  - たとえ：明日までに，社員1が，仕事Aと仕事Bをこなす

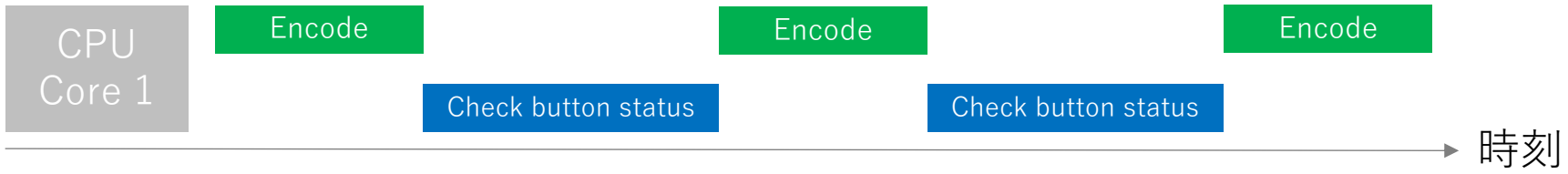


- 並列 Parallelism
  - 複数の処理を同時に実行する
  - マルチコア／マルチプロセッサでないと実現できない
  - たとえ：明日までに，社員1が仕事A，社員2が仕事Bをこなす



並行が有益なシーンの例

- インタラクティブシステム
  - ある処理を実行中でも，ユーザからの入力を受け付ける必要がある
  - 例：動画エンコード中でも，ユーザが中止ボタンを押せる

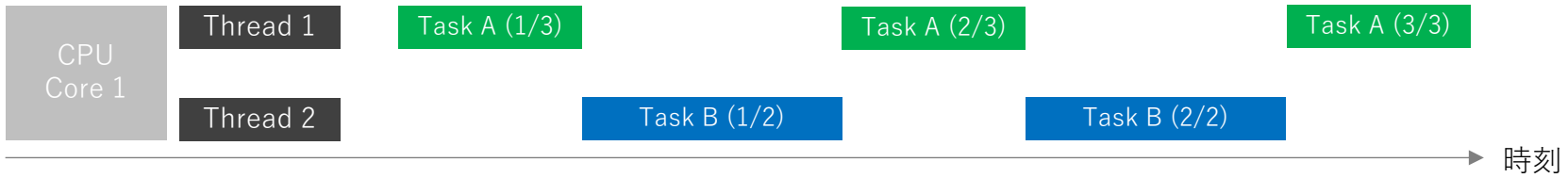


- 待ち状態がある処理を含むシステム
  - ある処理が待ち状態になった場合，他の処理を実行する
  - 例：外部サーバから返答を待っている間に，過去の返答内容を分析する

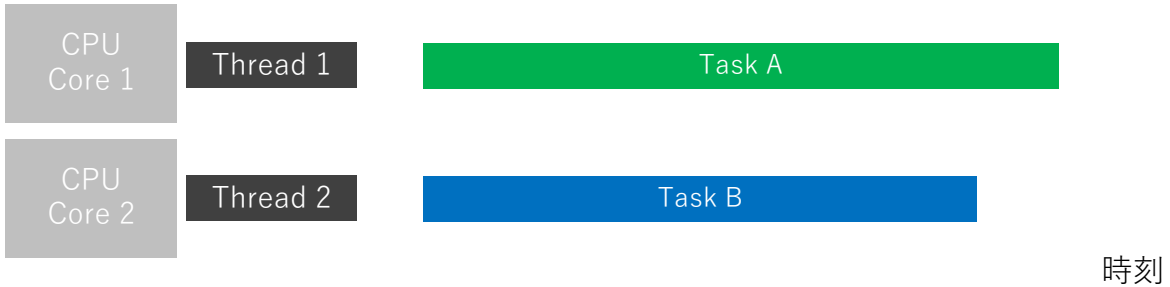
マルチスレッドと並行・並列

- マルチスレッド
  - 複数のスレッドが同時に実行状態にあること
  - 並行や並列を実現する手段

- 各スレッドが1つのCPUで動作している場合 → 並行



- 各スレッドが別のCPUで動作している場合 → 並列



各スレッドを明示的に別のCPUコアで動かす（＝並列）プログラミングは
実行可能環境が限られる上に難易度が高いため，以降，本講義では扱わない。

マルチスレッドプログラミング

マルチスレッドプログラミング

- 方法1：Threadクラスを継承する
- 方法2：Runnableインタフェースを実装する

作業準備

- 本日の作業ディレクトリの作成・移動
  - mkdir -p SOMEWHERE/2021_ap/07
  - 以降，SOMEWHERE/2021_ap/07をWORK_DIRとする

- 作業ディレクトリの作成・移動
  - cd WORK_DIR
  - Bb > 07: Multithreading 1 > Code > thread_test_1.zip をWORK_DIRにダウンロード
  - unzip thread_test_1.zip
  - cd thread_test_1

Threadクラスの継承による マルチスレッドプログラミング (1/4)

手順

[1] Threadクラスを継承したクラスを作成する
[2] そのクラス内でrun()をオーバーライドする
[3] そのクラスのインスタンス経由でstart()を呼び出す

MyThread1.java のアウトライン

```
                    [1]
public class MyThread1 extends Thread {
  [2]
    public void run() {
        // Do something.
    }
}
```

Main.java のアウトライン

```
public class Main {

    public static void main(String[] args) {
        MyThread1 thread = new MyThread1();
  [3]   thread.start();

        // Do something.
    }
}
```

これらの処理が
並行して行われる

Threadクラスの継承による マルチスレッドプログラミング (2/4)

MyThread1.java

```java
public class MyThread1 extends Thread {

    private final static long SLEEP_LEN_MSEC = 1000;


    public void run() {
        for(int i = 0; i < 10; i++) {
            System.out.println("MyThread1: " + i);
            try {
                Thread.sleep(SLEEP_LEN_MSEC);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

}
```

Main.java

```java
public class Main {

    private final static long SLEEP_LEN_MSEC = 1000;


    public static void main(String[] args) throws InterruptedException {
        MyThread1 thread = new MyThread1();
        thread.start();

        for(int i = 0; i < 10; i++) {
            System.out.println("Main: " + i);
            Thread.sleep(SLEEP_LEN_MSEC);
        }
    }

}
```

- run()
  Threadクラスで定義されているメソッド。
  サブクラスでオーバーライドして利用する。

- Thread.sleep()
  当該スレッドの処理を指定時間止めるメソッド。
  本質的には不要であるが，実行結果の視認性を高めるために便宜上利用している。

- InterruptedException
  一定時間スリープするThread.sleep()を実行する際に生じうる例外。
  MyThread1のrun()はThreadクラスからオーバーライドしたものであり，
  サブクラスで定義を変更してthrowsを付けることができないため，
  try・catchで例外処理を実装している。

```
% javac Main.java
% java Main
Main: 0
MyThread1: 0
Main: 1
MyThread1: 1
MyThread1: 2
Main: 2
. . .
```

実行するたびに表示順が異なる

Threadクラスの継承による マルチスレッドプログラミング (3/4)

MyThread1.java

```java
public class MyThread1 extends Thread {

    private final static long SLEEP_LEN_MSEC = 1000;

    public void run() {
        for(int i = 0; i < 10; i++) {
            System.out.println("MyThread1: " + i);
            try {
                Thread.sleep(SLEEP_LEN_MSEC);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

}
```

Main.java

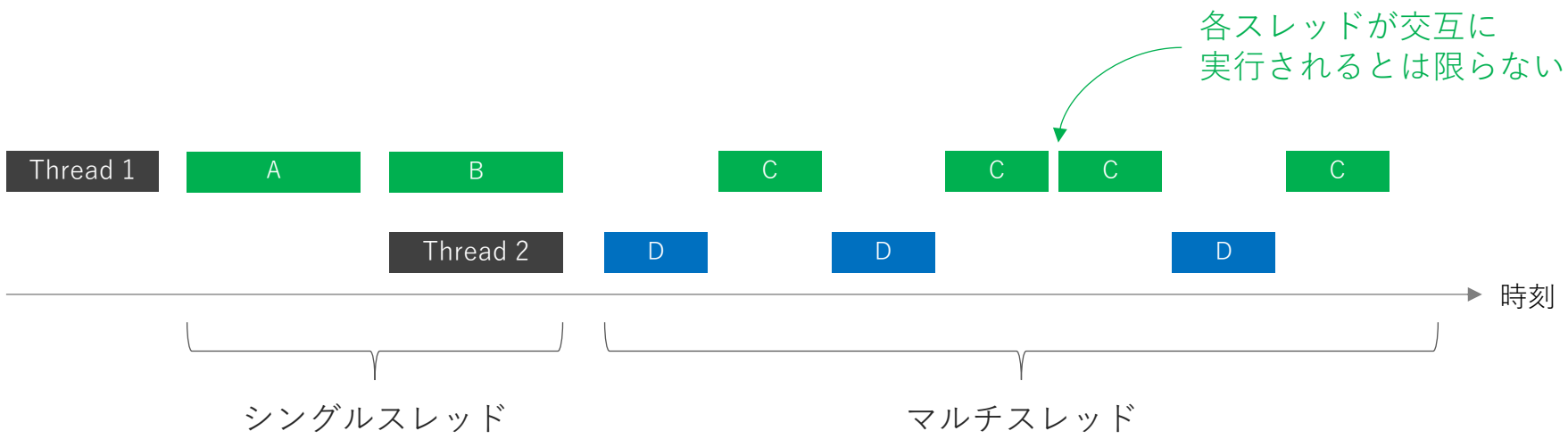
```java
public class Main {

    private final static long SLEEP_LEN_MSEC = 1000;

    public static void main(String[] args) throws InterruptedException {
        MyThread1 thread = new MyThread1();
        thread.start();

        for(int i = 0; i < 10; i++) {
            System.out.println("Main1: " + i);
            Thread.sleep(SLEEP_LEN_MSEC);
        }
    }

}
```



IR07-1

- 10本のスレッドが，それぞれ0〜9までの数字を1秒おきにコンソール上に出力するプログラムを作成せよ。

目標の出力

```
MyThread #0: 0
MyThread #1: 0
MyThread #3: 0
MyThread #2: 0
MyThread #6: 0
MyThread #5: 0
MyThread #4: 0
MyThread #7: 0
MyThread #8: 0
MyThread #9: 0
MyThread #0: 1
MyThread #5: 1
MyThread #2: 1
.
.
.
MyThread #5: 8
MyThread #3: 8
MyThread #0: 8
MyThread #2: 9
MyThread #9: 9
MyThread #4: 9
MyThread #8: 9
MyThread #6: 9
MyThread #1: 9
MyThread #7: 9
MyThread #3: 9
MyThread #5: 9
MyThread #0: 9
```

ヒント

- Threadクラスを継承するクラスのインスタンスを10個作る
  （当然だが10回同じコードを書いてはいけない）
- 上記インスタンス作成時に，スレッドのIDを与える
- 各インスタンス経由でstart()を呼び出す

Threadクラスの継承による マルチスレッドプログラミング (4/4)

- Threadクラスを継承する方法の問題点
  - Javaでは多重継承が許されていない（1つのクラスしか継承できない）ため Threadクラスを継承してしまうと他のクラスを継承できない
  - 公式ドキュメントにおいてもこの方法は相対的に一般的ではなく，制約があるので，もう一方の方法（後述の方法2）の利用を推奨している
    https://docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html

MyThread1.java

```java
public class MyThread1 extends Thread {

    private final static long SLEEP_LEN_MSEC = 1000;


    public void run() {
        for(int i = 0; i < 10; i++) {
            System.out.println("MyThread1: " + i);
            try {
                Thread.sleep(SLEEP_LEN_MSEC);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

}
```

マルチスレッドプログラミング

- 方法1：Threadクラスを継承する
- 方法2：Runnableインタフェースを実装する

作業準備

- 作業ディレクトリの作成・移動
  - cd WORK_DIR
  - Bb > 07: Multithreading 1 > Code > thread_test_3.zip をWORK_DIRにダウンロード
  - unzip thread_test_3.zip
  - cd thread_test_3

Runnableインタフェースの実装による マルチスレッドプログラミング (1/2)

手順

[1] Runnableインタフェースを実装したクラスを作成する
[2] そのクラス内でrun()を実装する
[3] そのクラスのインスタンスを引数にしてThreadクラスのインスタンスを作成する
[4] Threadクラスのインスタンス経由でstart()を呼び出す

MyThread3.java のアウトライン

```
                    [1]
public class MyThread3 implements Runnable {
  [2]
    public void run() {
        // Do something.
    }
}
```

Main.java のアウトライン

```
public class Main {

    public static void main(String[] args) {
  [3]   MyThread3 mt = new MyThread3();
        Thread thread = new Thread(mt);
  [4]   thread.start();

        // Do something.
    }
}
```

これらの処理が
並行して行われる

Runnableインタフェースの実装による マルチスレッドプログラミング (2/2)

MyThread3.java

```java
public class MyThread3 implements Runnable {

    private final static long SLEEP_LEN_MSEC = 1000;


    public void run() {
        for(int i = 0; i < 10; i++) {
            System.out.println("MyThread3: " + i);
            try {
                Thread.sleep(SLEEP_LEN_MSEC);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

}
```

Main.java

```java
public class Main {

    private final static long SLEEP_LEN_MSEC = 1000;


    public static void main(String[] args) throws InterruptedException {
        MyThread3 mt = new MyThread3();
        Thread thread = new Thread(mt);
        thread.start();

        for(int i = 0; i < 10; i++) {
            System.out.println("Main: " + i);
            Thread.sleep(SLEEP_LEN_MSEC);
        }
    }

}
```

```
% javac Main.java
% java Main
Main: 0
MyThread3: 0
Main: 1
MyThread3: 1
MyThread3: 2
Main: 2
. . .
```

実行するたびに表示順が異なる

IR07-2

- Threadクラスの継承を行わずに，10本のスレッドが，それぞれ0〜9までの数字を1秒おきにコンソール上に出力するプログラムを作成せよ。

目標の出力

```
MyThread #0: 0
MyThread #1: 0
MyThread #3: 0
MyThread #2: 0
MyThread #6: 0
MyThread #5: 0
MyThread #4: 0
MyThread #7: 0
MyThread #8: 0
MyThread #9: 0
MyThread #0: 1
MyThread #5: 1
MyThread #2: 1
.
.
.
MyThread #5: 8
MyThread #3: 8
MyThread #0: 8
MyThread #2: 9
MyThread #9: 9
MyThread #4: 9
MyThread #8: 9
MyThread #6: 9
MyThread #1: 9
MyThread #7: 9
MyThread #3: 9
MyThread #5: 9
MyThread #0: 9
```

作業準備

- 作業ディレクトリの作成・移動
  - cd WORK_DIR
  - Bb > 07: Multithreading 1 > Code > echo.zip をWORK_DIRにダウンロード
  - unzip echo.zip
  - cd echo

IR07-3

❖ Threadクラスの継承を行わずに，次の挙動を実現するプログラムを作成せよ。

- キーボードから1〜9の数字を1つ入力してEnterを押すと，
  その数字がコンソール上に繰り返し表示され続ける。
- 上記の数字が表示され続けている間に，新たに1〜9の数字を1つ入力してEnterを押すと，
  その数字がコンソール上に繰り返し表示され続ける。0を入力してEnterを押すとプログラムが終了する。

ベースとなるコード (Echo.java)

```java
import java.util.Scanner;

public class Echo {

    private final static long SLEEP_LEN_MSEC = 100;

    private Scanner scanner;
    private int n;


    public Echo() {
        scanner = new Scanner(System.in);
        System.out.print("n > ");
        n = scanner.nextInt();
    }

    private void echo() throws InterruptedException {

        while(n != 0) {
            System.out.print(n);
            Thread.sleep(SLEEP_LEN_MSEC);
        }

    }


    public static void main(String[] args) throws InterruptedException {
        Echo ste = new Echo();
        ste.echo();
    }

}
```

ベースとなるコードの挙動

```
% javac Echo.java
% java Echo
n > 1
1111111111111111. . .
```

一度whileループに入ってしまうと
他の処理は一切できないので
題意が満たせない。

❖ キーボードから数字を読み込む方法
Scanner scanner = new Scanner(System.in);
int i = scanner.nextInt();

❖ InterruptedException
一定時間スリープするThread.sleep()を実行する際に生じうる例外。

IR07-4

- 自身のIDと指定文字列をコンソール上に出力するprint()を持つPrinterクラスがある。このクラスを継承して，(1)の機能を持つSimplePrinterクラスと，(2)の機能を持つDecoratePrinterクラスを実装せよ。次に，(1)(2)の機能を並行して実行するプログラムを完成させよ。

  (1) 0～4の数字を順番に，2秒おきに，シンプルに出力する
  (2) 0～9の数字を順番に，1秒おきに，装飾して出力する

Printer.java

```java
public class Printer {

    private int id;


    public Printer(int id) {
        this.id = id;
    }


    public void print(String msg) {
        System.out.println("Printer #" + id + ": " + msg);
    }
}
```

目標の出力例（Printer #0がシンプル，Printer #1が装飾）

```
Printer #1: =====0=====
Printer #0: [0]
Printer #1: =====1=====
Printer #0: [1]
Printer #1: =====2=====
Printer #1: =====3=====
Printer #0: [2]
Printer #1: =====4=====
Printer #1: =====5=====
Printer #0: [3]
Printer #1: =====6=====
Printer #1: =====7=====
Printer #0: [4]
Printer #1: =====8=====
Printer #1: =====9=====
```

本日のまとめ

❖講義内容

- Threadクラスの継承によるマルチスレッドプログラミング
- Runnableインタフェースの実装によるマルチスレッドプログラミング

❖授業内課題提出

- 各授業内課題（IR）の解答を記載せよ。
- 「講義内容のまとめ」の解答欄に
  上記「講義内容」の各項目について文章で説明を記載せよ。