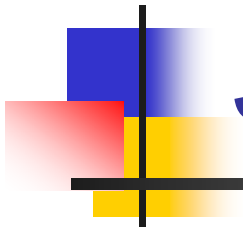
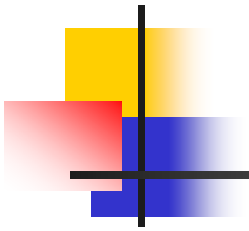


Javaゼミ第九回目

9.1 スレッドの概要

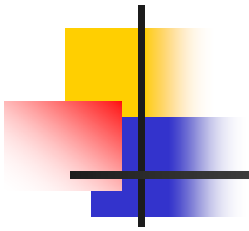
9.2 スレッドの作成

発表日　6月14日　　発表者　加藤 友宏



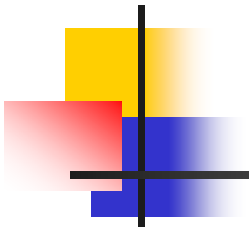
スレッド概要

- OSは、1台のコンピュータ上で複数のプロセスを管理。
- 各プロセスは、固有のアドレス空間で動作して、独立したスケジュールで実行される。
- スレッドとは、プロセス内の実行の流れ。
- スレッドは、所属するプロセスのメモリ、その他リソースを使う　→　プロセスで使うより少ないリソースでよくなる。



スレッドの概要

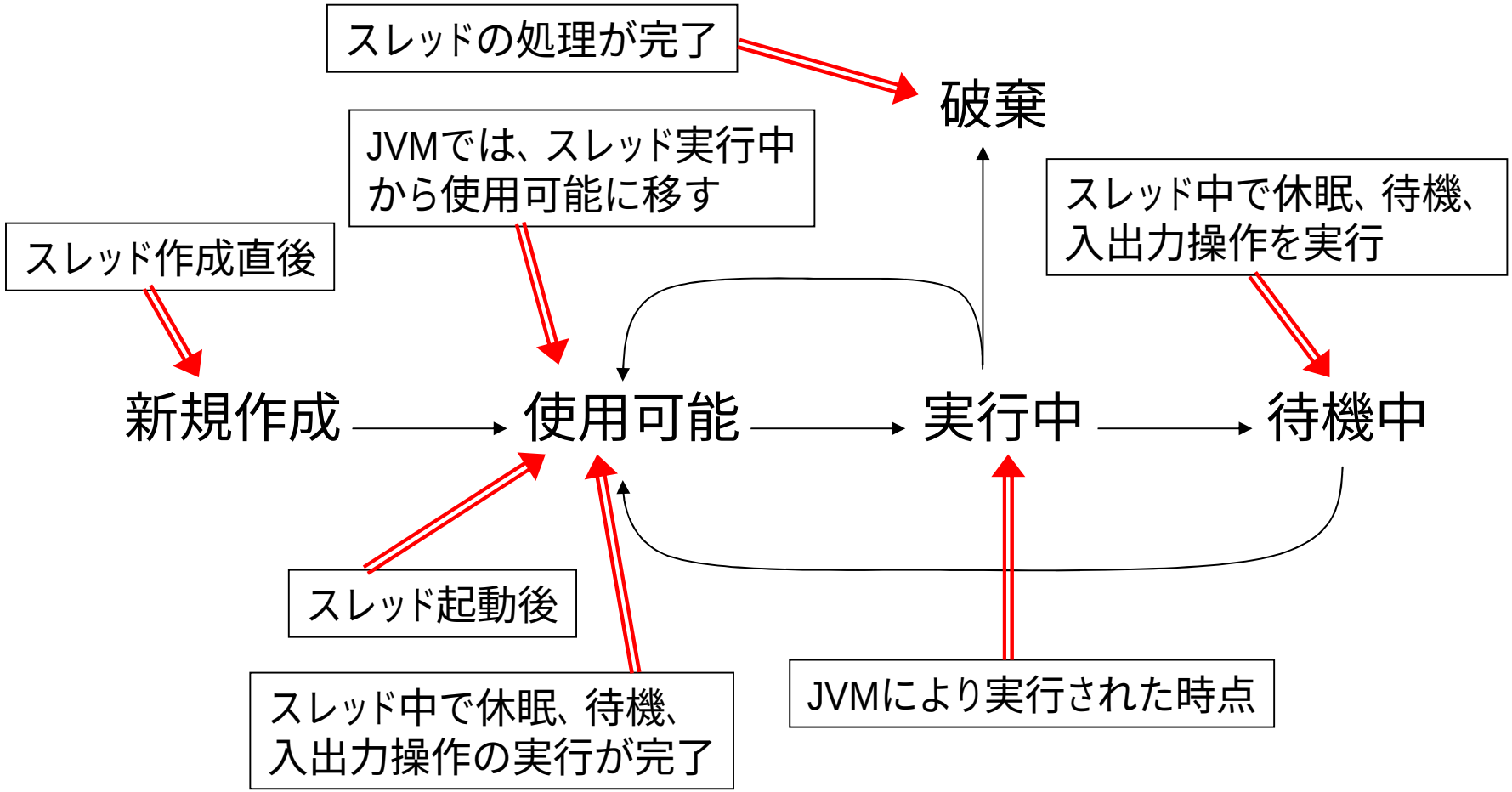
- プロセスでは、複数のスレッドの使用が可能
- JVMでは、スレッドを管理、実行スケジュールを決定
- スレッドの実行コンテキストの切り替えは、プロセスの切り替えに比べ、格段に短くなる

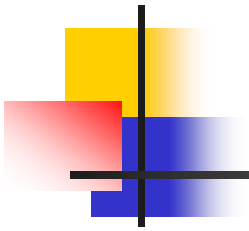


スレッドの概要 (スレッドの一生)

- スレッドの状態は、新規作成、使用可能、実行中、待機中、破棄の5種類。
- 作成直後に新規作成、起動後に使用可能になる。
- JVMに実行された時点で実行中になる。
- スレッドの処理が完了すると破棄状態、遷移する。
- スレッドで休眠、待機、入出力操作のいずれかの実行を選択すると待機中になる。
- 休眠、待機、入出力操作の実行完了後、使用可能になる。
- JVMでは、スレッドを実行中から使用可能に移すので、ほかのスレッドに実行のチャンスが巡ってくる

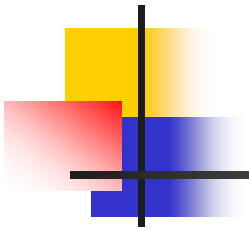
スレッドの概要 (スレッドの一生)





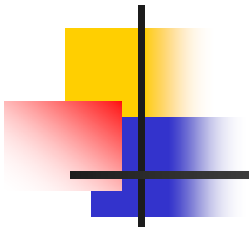
スレッドの概要 (優先順位)

- スレッドには、優先順位がある。
- JVMがスケジュールの管理をする。
- 使用可能なスレッドが複数あるとき、その中で優先順位の最も高いものを選択し実行。
- 使用可能な優先順位が高いスレッドがあるとき、優先順位が低いスレッドは実行されない。
- 同じ優先順位のスレッドのときは、JVMがCPU処理を時分割する。



スレッドの概要 (優先順位)

- 一定の時間が経過後、実行中のスレッドを強制中断し、ほかのスレッドに実行チャンスを与える。
- 優先順位の同じスレッドにCPU処理が平等に割り当てられる。
- ただし、すべてのJVMに時分割機能があるとは言えない。
- 時分割ができない時は、実行されているスレッドよりも優先順位の高いスレッドが使用可能にならない限り、強制的に中断されない。



スレッドの作成(Threadクラス)

- スレッドの作成と管理は、java.langパッケージのThreadクラスを使う。
- スレッドは、どれもThreadクラスの独立したインスタンス。

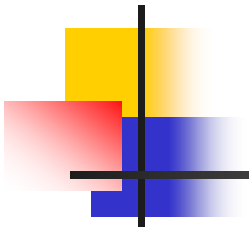
スレッドの作成(Threadクラス)

- Threadクラスの拡張

```
class ThreadX extends Thread {
  public void run(){
      //スレッドの処理
  }
}
```

- ThreadをThreadXクラスが拡張。
- スレッドの処理は、run()メソッドの中に記述。
- ほかのオブジェクトやメソッドを作成することもできる。

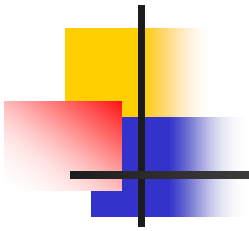
スレッドの作成(Threadクラス)



- スレッドの起動

```
ThreadX tx = new ThreadX();
tx.start();
```

- ThreadXクラスをインスタンス化する。
- オブジェクトのstart()メソッドを呼び出しスレッドの実行を開始する。
- start()メソッドでは、run()メソッドの呼び出しも行われる。
- 複数のThreadXクラスのインスタンスを作成、起動し、同時に実行することも可能



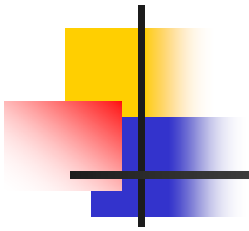
スレッド作成の例(Threadクラス)

- Threadクラスのサブクラスを使ってスレッドを作成したプログラム

```
class ThreadX extends Thread {
  public void run() { // ThreadXのスレッド内容
        System.out.println("OK1");
  }
}

class ThreadDemo {
  public static void main(String args[]) {
    ThreadX tx = new ThreadX(); // ThreadXクラスからtxオブジェクトを生成
    tx.start(); // txオブジェクトを開始
  }
}
```

スレッドの作成(Runnable)



- Runnableインターフェイスを実装したクラスを宣言する方法もある。
- run()メソッド

```
public void run();
```

- クラスの宣言

```
class RunnableY implements Runnale {
  public void run() {
    //スレッドの処理
  }
   :
}
```

スレッドの作成(Runnable)

- スレッドの起動

```
RunnableY ry = new RunnableY();
Thread ty = new Thread(ry);
ty.start();
```

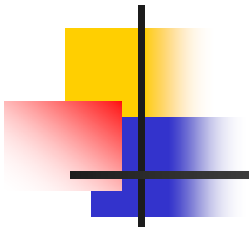
- 一行目で、RunnableYクラスをインスタンス化
- 二行目で、Threadクラスをコンストラクタへの引数として、RunnableYオブジェクトへの参照を引き渡しインスタンス化。
- 最後の行でスレッドの起動

スレッドの作成例(Runnable)

- Runnableインターフェイスを実装してスレッドを作成した例

```
class RunnableY implements Runnable {
  public void run() { // スレッド内容
        System.out.println("OK2");
  }
}

class RunnableDemo {
  public static void main(String args[]) {
    RunnableY ry = new RunnableY(); // RunnableYオブジェクトをインスタンス化
    Thread ty = new Thread(ry); // Threadオブジェクトをインスタンス化
    ty.start(); // tyオブジェクトを開始
  }
}
```

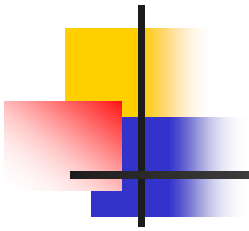


スレッドの作成

- Threadクラスには、スレッドの優先順位を表すint型の定数が3つ定義されている。
- MAX_PRIORITY ・・・ 最も優先順位が高い
- MIN_PRIORITY ・・・ 最も優先順位が低い
- NORM_PRIORITY ・・・ 通常の優先順位
- Threadコンストラクタ

```
Thread()
Thread(Runnable r)
Thread(Runnable r, String s)
Thread(String s)
```

・rは、Runnableインターフェイスを実装したオブジェクトへの参照
・sは、スレッドを表す文字列



スレッドの作成

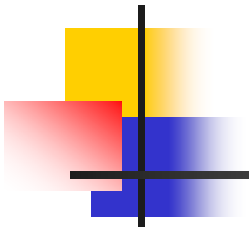
- Threadクラスで定義されている主な静的メソッド

メソッド	説明
static Thread currentThread()	現在のスレッドへの参照を返す
static void sleep(long msec) throws InterruptedException	現在のスレッドをmsecミリ秒間、待機させる
static void sleep(long msec, int nsec) throws InterruptedException	現在のスレッドをmsecミリ秒+nsecナノ秒間、待機させる
static void yield()	現在のスレッドからほかのスレッドにCPU処理の制御を明け渡す

スレッドの作成

- Threadクラスに定義されている主なインスタンスメソッド

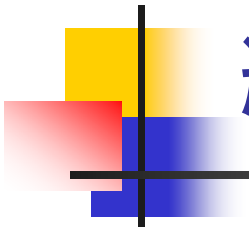
メソッド	説明
String getName()	スレッドの名前を返す
int getPriority()	スレッドの優先順位を返す
void interrupt()	スレッドを中断する
boolean isAlive()	スレッドが開始され、まだ破棄されていない場合に真を返す。それ以外の場合は偽を返す。
boolean isinterrupted()	スレッドが中断されている場合に真を返す。それ以外の場合は偽を返す。
void join() throws interruptedException	このスレッドが破棄されるまで、呼び出し元を待機させる



スレッドの作成

- Threadクラスに定義されている主なインスタンスメソッドの続き

メソッド	説明
void join(long msec) throws interruptedException	このスレッドが破棄されるまで、最大msecミリ秒間まで呼び出し元を待機させる。msecが0の場合、待機時間に制限はない。
void join(long msec, int nsec) throws interruptedException	このスレッドが破棄されるまで、最大msecミリ秒+nsecナノ秒間まで呼び出し元を待機させる。msecとnsecがどちらも0の場合、待機時間に制限はない。
void run()	スレッドの処理を構成する。このメソッドはサブクラスでオーバーライドされる。
void setName(String s)	スレッドの名前をsにする。
void setPriority(int p)	スレッドの優先順位をpにする。
void start()	スレッドを開始する。
String toString()	このスレッドに相当する文字列を返す。



練習問題1

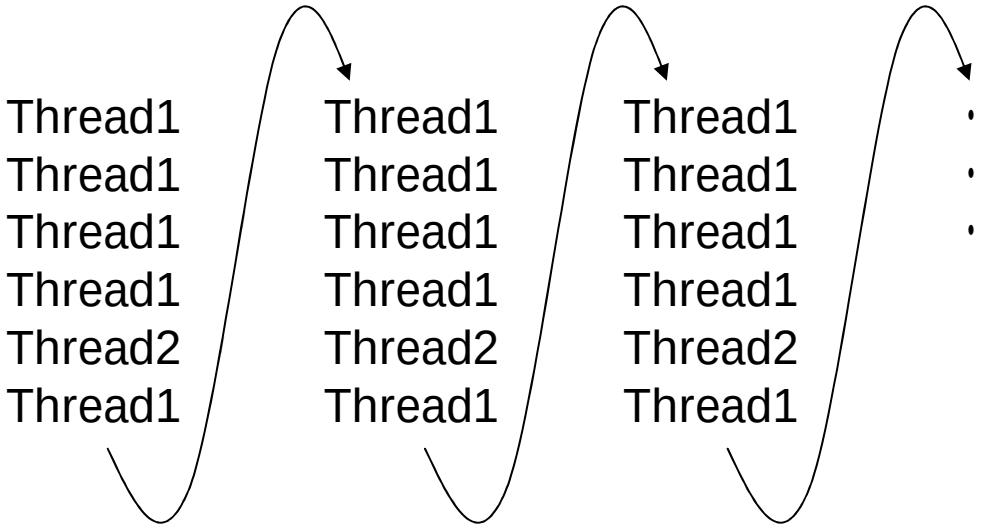
- 2つのスレッドを実行するプログラムを作成せよ。1つ目のスレッドは、1秒ごとに、

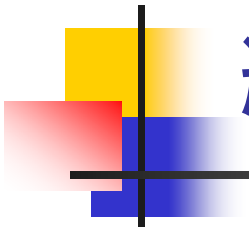
  ”Thread1”を表示し、2つ目のスレッドは、5秒ごとに”Thread2”を表示するものとする。スレッド名は任意。

  なお、Threadクラスを拡張して、スレッドを作成すること。

練習問題2

- 問題1で作成したプログラムのスレッドを
  Runnableインターフェイスを実装して作成せよ。
- 問題1と問題２の実行結果





練習問題3

- 3つのスレッドを実行するプログラムを作成せよ。
  1つ目のスレッドは、3秒ごとに、”Thread1”を
  表示してから、2つ目のスレッドを開始する。
  2つ目のスレッドは、”Thread2”を表示してから、
  3つ目のスレッドを開始する。
  3つ目のスレッドは、”Thread3”を表示する。