

インターネットプログラミング
2401教室
第1回
2014/9/17

岩井将行

授業資料

- <http://www.cps.im.dendai.ac.jp>

講師紹介

- <http://cps.im.dendai.ac.jp/index.php?Members%2Fiwai>
- 岩井研究室
- <http://cps.im.dendai.ac.jp>
- 岩井研究室の研究分野
- <http://cps.im.dendai.ac.jp/index.php?Research%2FTopics>
- 連絡先 1号館11F 11107b
- iwaiあつと im.dendai

TA・SA・副手

- たじまっち
- こんどうこん
- しの
- 他岩井研の精鋭

成績

- 毎回取り組み姿勢(出席)
 - 毎回課題(演習のみ)
 - 中間試験(座学)
 - 最終試験(座学)
 - 最終課題(演習のみ)
-
- ★演習は演習最終発表会を加味

講義内容

[第1回](#) Java理解度チェック

[第2回](#) Javaプログラミング基礎2

[第3回](#)

TCP/IPの復習 TCPサーバ

[第4回](#)

TCPクライアント/サーバ通信 チャットプログラム

[第5回](#)

UDP通信

[第6回](#)

中間学力考査（持ち込み不可 紙提出）

[第7回](#)

スレッド基礎 サーバのスレッド化 マルチスレッド

[第8回](#)

デザインパターン

ファクトリメソッド シングルトン

[第9回](#)

ノンブロッキングI/O Javaプログラミング応用

[第10回](#)

マルチスレッド スレッドプール

[第11回](#)

Webクライアント

[第12回](#)

WEBサーバ,プロジェクト設計

[第13回](#)

プロジェクト実装1

[第14回](#)

プロジェクト実装2

20 [第15回](#)

学力考査（持ち込み可 プログラミング提出）

授業予定日日程

- http://www.soe.dendai.ac.jp/kyomu/portal/2014_schedule_t.pdf
- スケジュール +
- (1)9/17 第1回 Java理解度チェック
- (2)9/24 第2回 Javaプログラミング基礎1
- (3)10/1 第3回 Javaプログラミング基礎2 TCP/IPの復習
TCPサーバ
- (4)10/8 第4回 TCPクライアント/サーバ通信 チャットプログラム
- (5)10/15 第5回 UDP通信
- (6)10/22 第6回 中間学力考査（持ち込み不可 紙提出）

概要

- クライアント／サーバモデル、TCP/IPネットワークのアプリケーションプログラミングインタフェースの基本および、ネットワークアプリケーションを効率的に動作させるためのマルチスレッドプログラミングを講義する。この基本の後、チャット等の対話型アプリケーション、Twitter4J等のアプリケーション開発の実例を講義する。

ゴール

- 通信ネットワークを利用したアプリケーションソフトウェアを、TCP/IP を意識したレベルで作成できる力を養成することを目標とする。

参考書

- 購入の必要はありません。
 - TCP/IPソケットプログラミング JAVA編
 - ISBN4-274-06520-0
 - オーム社
 - •[改訂第2版]JAVA言語プログラミングレッスン上
 - ISBN4-7973-3211-5
 - SoftBankCreative
 - •[改訂第2版]JAVA言語プログラミングレッスン下
 - ISBN4-7973-3212-3
 - SoftBankCreative

持ち物

- Laptop pc
 - Eclipse環境が整っていること
- PDF アクロバトリーダ
- Wifiでネットワークに接続できること。
- Macでもよい

Java理解度チェック

今日のトピック

- コンパイル
- 変数と型
- If/switch
- For/while
- 配列
- クラスとインスタンス
- コンストラクタとメソッド

Javaプログラミング

- プログラミングとは？
- プログラムを作るための方法
- プログラムとは？
- 決まった手順を決まったように実行
- するようにコンピュータに渡す命令群
- 信号機プログラム
- 信号を赤にしろ→30秒まで→信号
- を青にしろ→ 30秒まで→信号を
- 黄色にしろ→ 3秒まで→(最初にも
- どれ)

プログラム

- 命令として与えられている以外のことは一切しません。
- 命令は細かく与える必要があります。コンピュータは人間のように賢くありません。細かく順序正しく指示しないと
• いけません。
- コンピュータは命令を忠実に最後までやり遂げます。もし、*1万回同じことを繰り返しやるように指示されると、コンピュータはその通り1万回繰り返します。*
- それぞれの命令は単純ですが、それを非常に速く行うことができます。

ファイルの種類について

ソースファイル・ソースコード

- Java言語で書かれたプログラムファイル
- *.java

クラスファイル 中間コード

- コンパイラによって生成されたファイル
- *.class

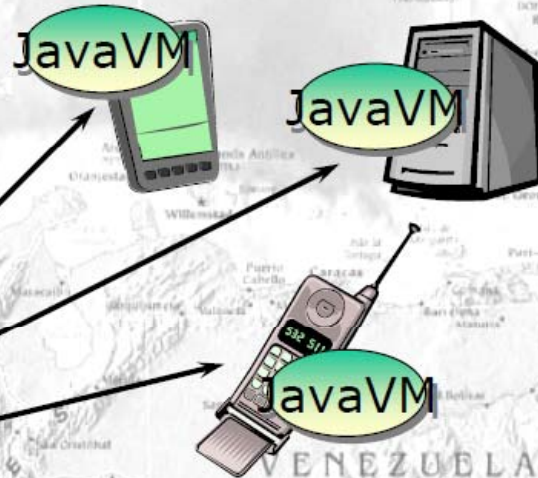
```
import org.w3c.dom.*;  
import org.xml.sax.*;  
import javax.xml.parsers.*;  
  
public class ChessboardDOMPrinter {  
    private DocumentBuilder builder;
```

ソースコード

コンパイル作業

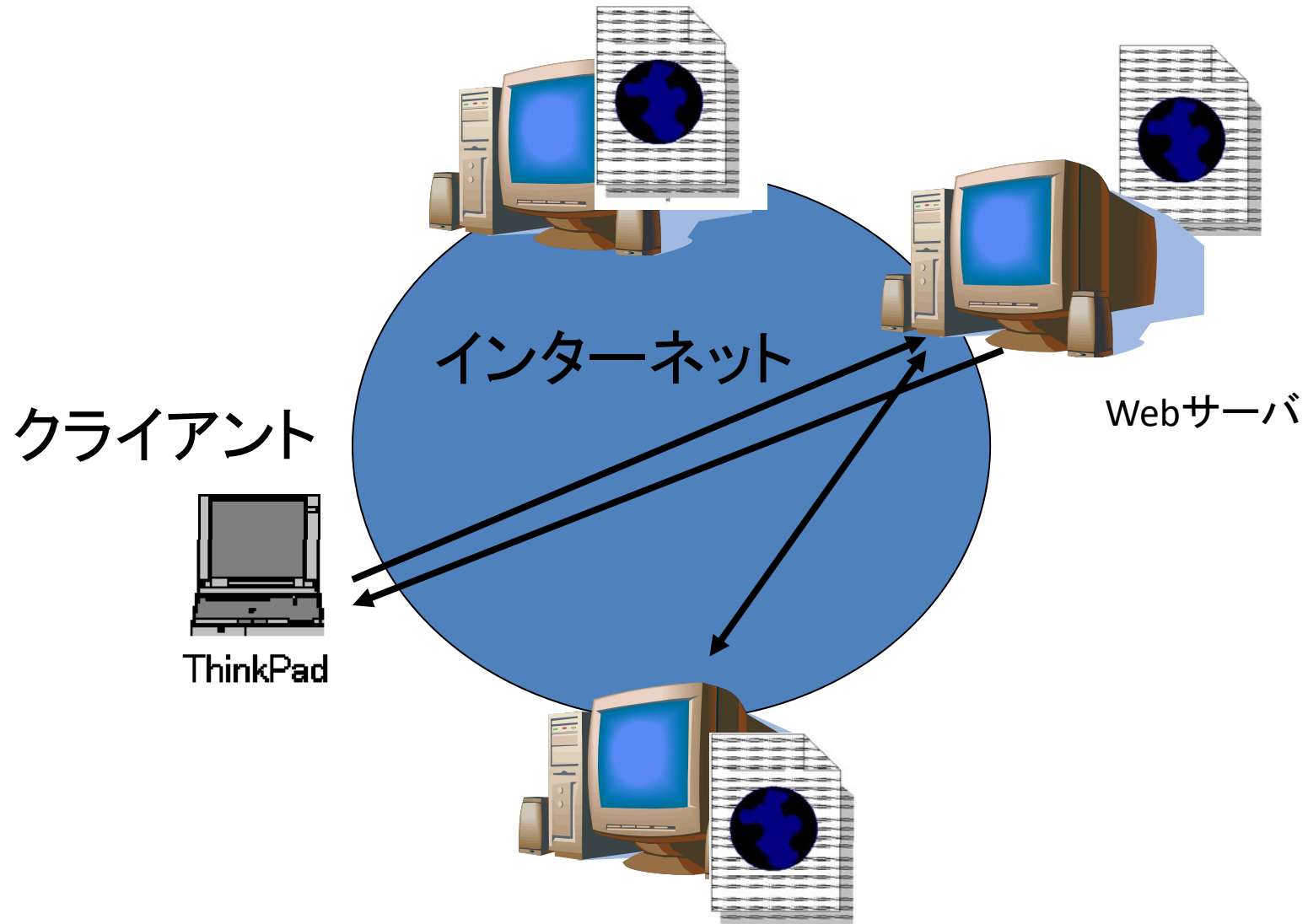
```
XOXXXX  
XOXXXX  
XOXXXX  
XOXXXX
```

中間コード



機械に応じた命令後

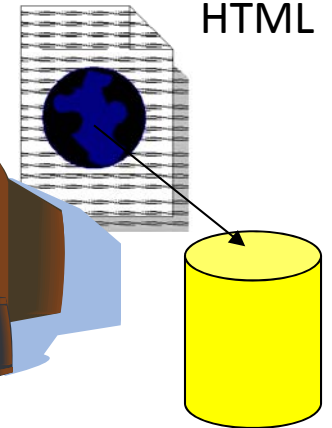
Webアプリケーション全体像



Java アプレット

Applet =
“Application” + “-let”

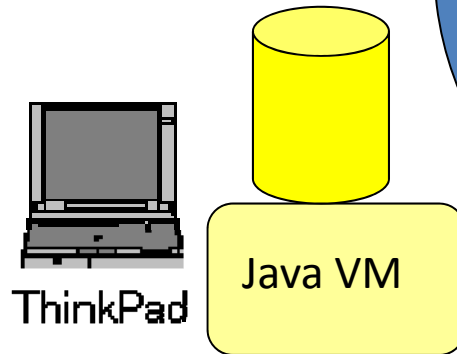
Webサーバ



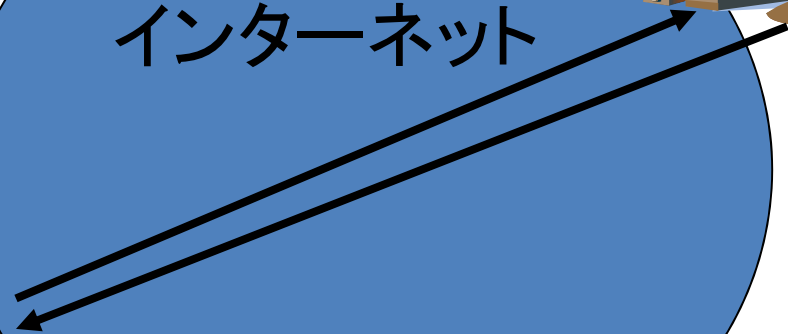
Weather.class

インターネット

クライアント



クライアントで実行

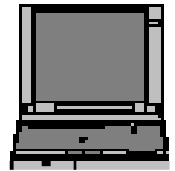


Java サーブレット

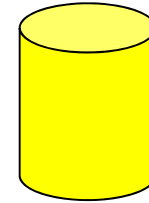
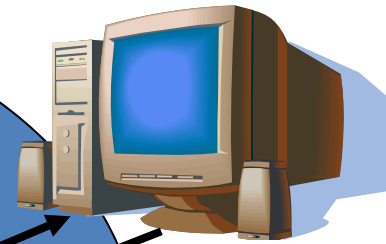
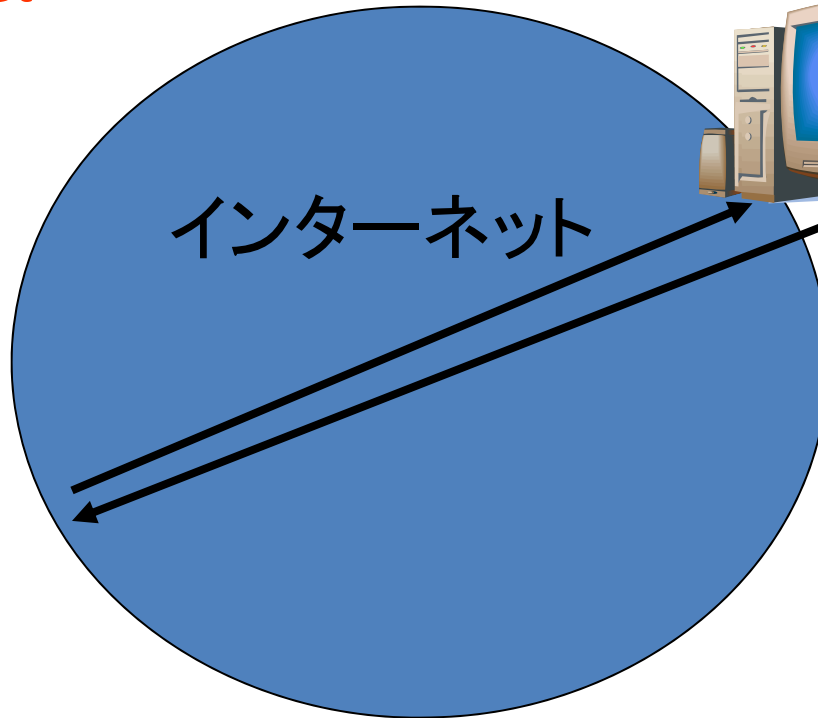
Servlet =
“Server” + “-let”

Webサーバ

クライアント



ThinkPad



Account.class

サーバで実行

JDK

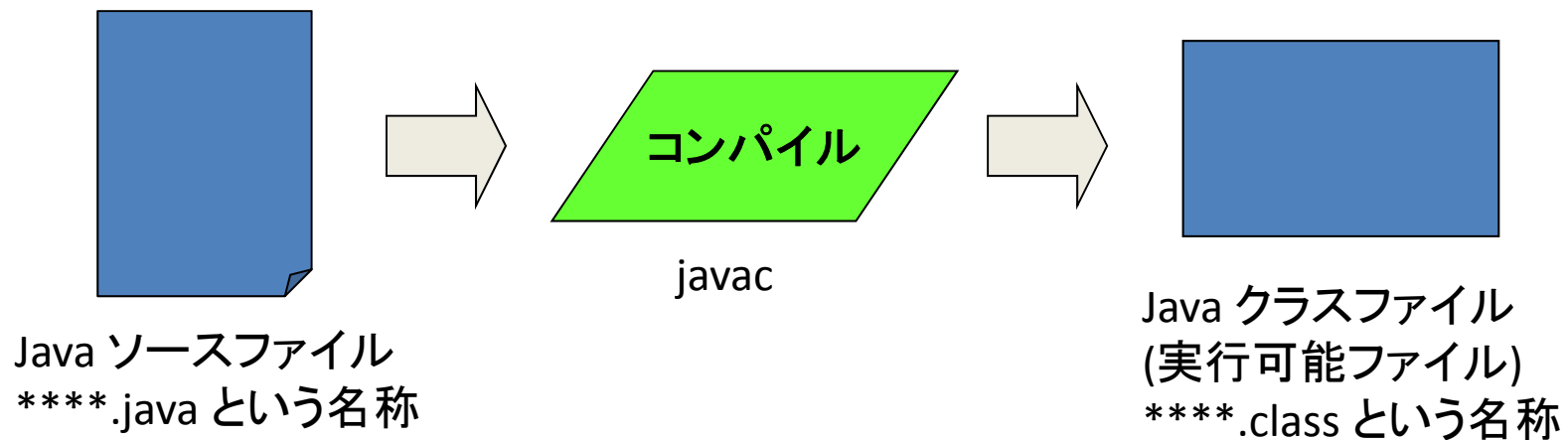
- JDKはJava Developers Kitの略
- Javaの開発環境として提供
- Javaでの開発のために必要な、コンパイラなどのツール群
- Sunのホームページから無償でダウンロードが可能
- JDKの構成
 - Javaインタプリタ(java)
 - Javaコンパイラ(javac)
 - Javaデバッガ(jdb)
 - 逆アセンブラ(javah)
 - C言語コードへの変換ツール(javah)
 - ドキュメント生成ツール(javadoc)
 - JDK バージョンからの移行ツール(upgrade)
 - 標準クラス・ライブラリのパッケージ群
- JDKによる開発手順
 - (1) ソース・ファイルのエディット
 - (2) javacによるコンパイル
 - (3) 生成されたバイト・コードの実行

コンピュータ教室での環境

- エディタ TeraPad
- コマンドプロンプト上で、
 JDKコマンドを動作させる

コンパイル

- 人間が書いたプログラムをコンピュータが理解できるような状態に変換
- 正常にコンパイルできると****.classというファイルができる
- コンピュータが理解できる状態になったプログラムがJavaクラスファイル
- コンパイルするときに, javac というプログラム(コンパイラ)を使用

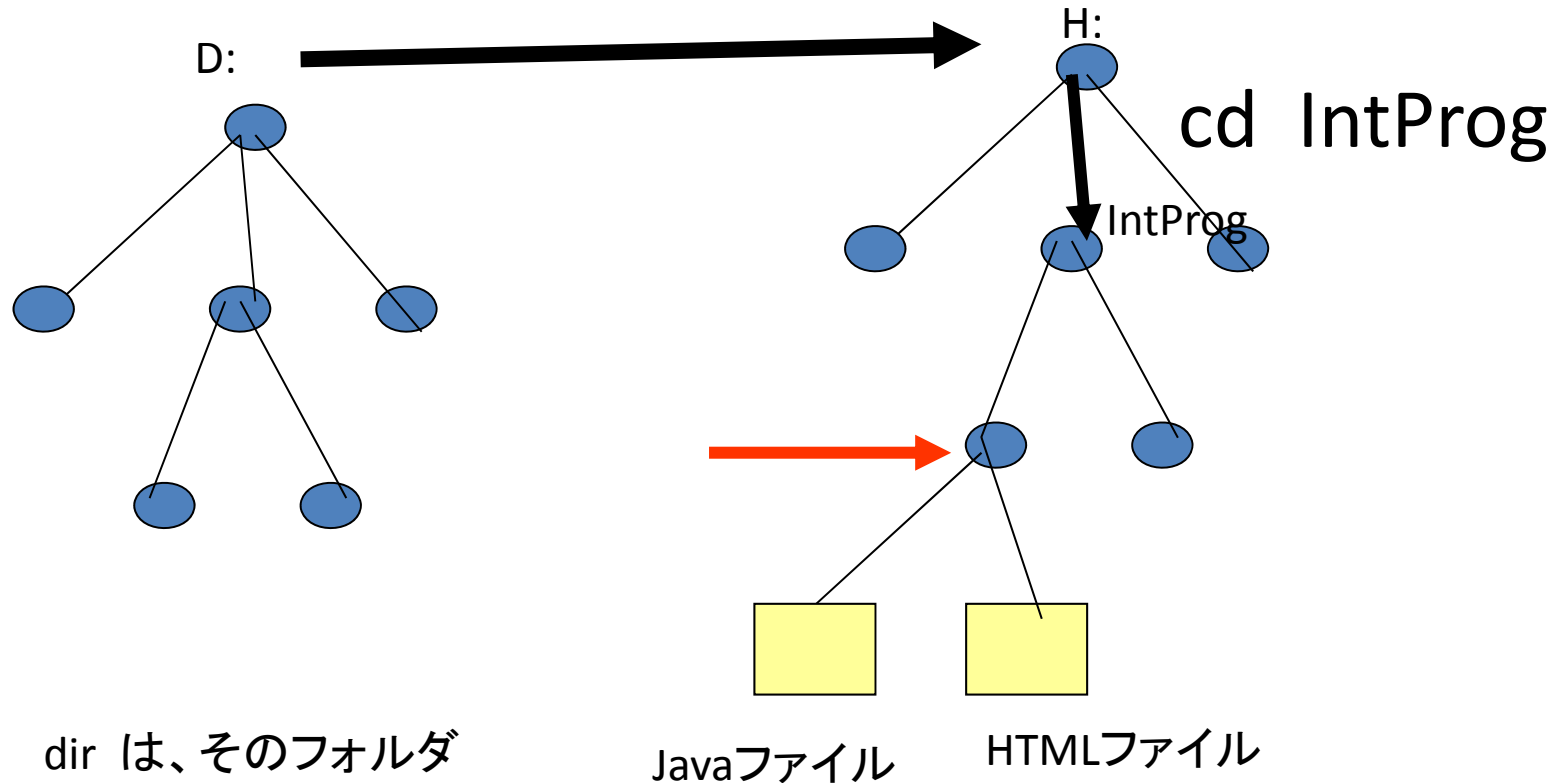


コンパイルの実際

- (1) コマンドプロンプトを開く。
- (2) H:
- (3) cd Java
- (4) dir で、Hello.java
があることを確認する。
- (5) javac Hello.java
- (3) エラーがなければ、Hello.class が生成
される。
- (4) エラーメッセージがあれば、修正する。

フォルダ=ディレクトリ

H: とだけ打つ。



dir は、そのフォルダ
にあるファイルのリスト
を表示させるコマンド

プログラムの実行

```
java Hello
```

- Hello.java を編集して表示される文字を変えてみよう

Hello.java

```
public class Hello {  
  
    public static void main(String[] args) {  
  
        System.out.println("Hello! ¥"木村君¥" ¥n ");  
        System.out.print("A");  
        System.out.print("B");  
        System.out.print("C");  
    }  
}
```

出力

```
>java Hello
```

```
Hello! "木村君"
```

```
ABC
```

Java プログラムの基本

クラス

```
public class Hello {
```

```
    public static void main(String[] args){
```

```
        // この中に、処理内容を書きます
```

```
    }
```

```
}
```

メソッド

注意1: Hello の部分がプログラム名

注意2: String は、文字列。String[] 文字列の配列

注意3: args は、コマンド引数の配列

args[0], args[1]

引数

```
>java Hikisu one two three
```

One

Two

Three

args[0]

args[1]

args[2]

引数を表示するプログラムを作ろう

```
public class Hikisu {  
  
    public static void main(String[] args) {  
        ???????  
  
    }  
}
```

基本型

- boolean 論理型 (true または false)
- char 整数型(文字型) (0以上65535以下)
- byte 整数型 (-128から127まで)
- int 整数型 (符号付き32ビット)
- long 整数型 (符号付き64ビット)
- float 実数 (単精度浮動小数点型)
- double 実数 (倍精度浮動小数点型)

演算子

- +
- -
- *
- /
- % (余り)

次の計算を答えを表示する プログラムを作ろう

- $3 + 5$
- $18 - 7$
- 32×5
- $10 \div 2$
- $300 \div 12$ の余り

次の計算を答えを表示する プログラムを作ろう

```
public class Calc1{  
  
    public static void main(String[] args) {  
        ???????  
  
    }  
  
}
```

String型を int 型に変換する方法

```
int abc = Integer.parseInt(String str)
```

```
String str_a = "12";
```

```
String str_b = "5"
```

```
int ans;
```

```
ans = Integer.parseInt(str_a) + Integer.parseInt(str_b)
```

引数に入れた2つの数字を加算するプログラムを作ろう

```
public class Calc2{  
  
    public static void main(String[] args) {  
        ???????  
  
    }  
  
}
```

コマンド入力読み込み

- Calc3.java
- reader をオブジェクトを作って、
reader.readLine(); メソッドで読み込む。
- BufferedReader クラスのオブジェクトを作る

コマンドラインでjavac

- Javaコンパイラーにパスが通っているか確認
- > javac -version

自分の環境でコンパイルして実行みよう

```
public class Hello{  
    public static void main(String[] args){  
        System.out.println("Hello");  
    }  
}
```

実行してみよう

- ファイル一覧を取得 *.classファイルがあるか
確認
- >dir
- 実行
- >java Hello

変数と値

- 変数: 値を入れておくもの
 - 変数の作成(変数宣言)
 - 値を入れる(代入)
 - 値を見る(参照)
- 型: どのような種類の値か
 - int
 - Char
 - string

If/switch

- 条件分岐
 - If(条件){处理}
 - If(条件){处理}else{处理}
 - If(条件){处理}else if(条件){处理}else{处理}
 - Switch(式){
 - Case 定数式:
 - 处理
 - break;
 - Default:
 - 处理}

for/while

繰り返し

- for(初期化; 条件式; 次の一步){繰り返す処理}
- while(条件式){繰り返す処理}

配列

- 複数の変数を番号をつけてまとめる
 - 宣言
 - 型名[] 配列名 = new 型名[要素の個数]
- –代入
 - ten[0]=34;
- –参照
 - ten[1]

クラスとインスタンス

- クラス
- 一般的な設計書
- インスタンス
- 実体

ネジクラスの実装

```
Public class Neji{
    double length; //ネジの長さ
    int pitch; //ピッチ幅
    String material; //材質
    public Neji(double l, int p, String m){
        length=l; pitch=p; material=m;
    }
    public String Sound(){
        for(int l=0; l < (int)length*10/10; i++){
            System.out.println("ギョツ");
        }
    }
}
```

インスタンス

- ネジクラスの実体化
 - 材質をどうするか
 - ピッチ幅はどうするか
- クラス名 インスタンス名 = new クラス名(インスタンス引数);
 - Neji n1 = new Neji(2.5, 1, “ステンレス”);
 - ステンレスで出来た2.5mmでピッチ幅1mmのネジ(n1)が出来る

コンストラクタとメソッド

- コンストラクタ
 - クラスの初期化処理をまとめたもの
- メソッド
 - 処理をまとめたもの

Static修飾子

- Newによってインスタンスを生成
 - インスタンスのメソッドやフィールド(変数)が使用可能となる
- Static修飾子の付いているメソッド/フィールド
 - プログラムの起動時にインスタンス化される
 - Newせずとも使用可能
 - Mainメソッド
 - `public static void main(String args[]){}`

本日の課題

- AさんとBさんとCさんのテストの点数を入力し平均点を出力するプログラムをつくる。
- ヒント1（文字列連結）
- Calc1.java
- ヒント2（二つの数をたす）
- Calc2.java

- クラス名: Calc3