

Java演習

第1回 (2024/4/10)

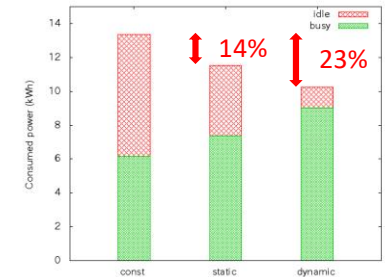
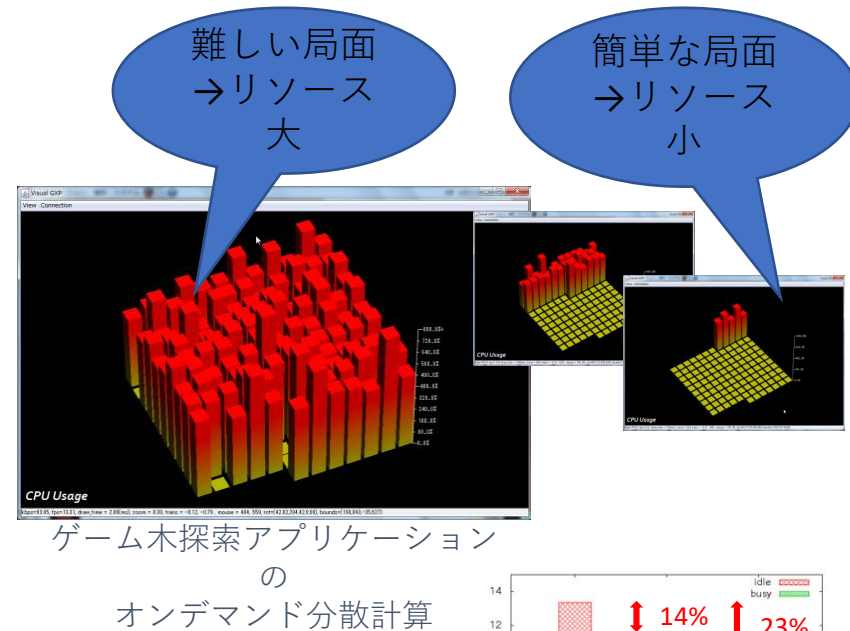
横山大作

自己紹介

横山大作

知的情報処理システム研究室

- ビッグデータ処理
- 並列・分散計算プログラミング環境
- ゲーム情報学
 - 激指 (コンピュータ将棋プレイヤ)



総消費電力の削減効果

講義概要

- 目標: Java言語を理解する
 - Javaの一通りの機能が使えるようにする
 - 比較的モダンな機能も
 - オブジェクト指向の概念に触れる
 - デバッグができるようになる
- 本講義の進め方
 - 評価は小課題（複数回）と期末試験の合計

講義予定

- 超高速でまず演習に参加できるだけの内容を押さえる！
 - 3回目までで高速入門完了
 - C言語とほぼ同じところや、似た概念の部分まで
 - 4回目で簡単なオブジェクト指向の利用まで
- その後、もう一度じっくり、説明飛ばしたところを押さえつつJava言語を見ていく
 - Javaらしいところは5回目以降でじっくりと
 - クラス、継承
 - コレクション



成績評価

- 割と毎回課す演習課題（小課題） 30%と期末試験の結果70%で成績評価を行う
 - 小課題は内容にコメントを（なるべく）付けて返しますが、原則「出したかどうか」だけの点数になります（出席点相当）
 - 何もソースを書き換えずに提出したとか、まったく解答の努力をしていないな、という場合には減点はします
 - 小課題だけでは単位は取れないが、期末試験だけでも良い成績は取れないのでそのつもりで。
 - 計算上、小課題を全部出しても期末テストでは $100 \times 3/7 = 42$ 点くらいは取らないと単位が来ないはず...理論上は。
 - 必修課題であることはある程度考慮しますが、あまりに期末の点数が低いと単位出せません
- 演習課題の提出、レポート提出はOh-o! Meiji

おやくそく

- 他人のプログラムをコピーすること、させることは不正行為である。不正行為が確認された場合には、単位が取り消されたり、定期試験での不正行為における処分と同等の処分が科せられることを十分認識すること

言語学習ですから

- 使ってなんぼです
- 身に着けてください
- 「データ構造とアルゴリズム実習」を活用してください
 - Javaを利用した演習授業になります
- 学習方法はいろいろ
 - ネットで検索、相談、聞く、本を読む...
- 人のを写すのは極めて効率悪い
 - 特にマウスでコピーするだけなら

進め方

- 講義の最後、20～30分程度は演習時間にするつもり
 - 小課題を解くための時間
 - 小課題は（わかっていれば）30分くらいで片付けられることを想定している
- ノートPCがない人も居るので、教室に居ることが必須ではない
- 横山は居るので、なるべくその時間に手を付けて、わからないところがあれば聞いてほしい

質問

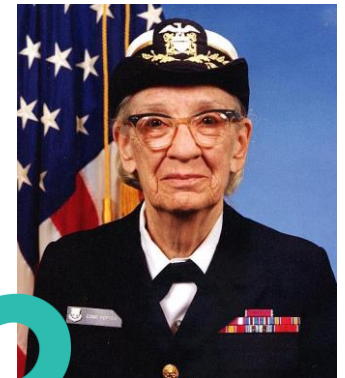
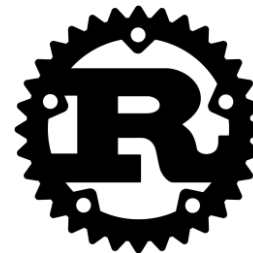
- パソコン持ってますか？
- Cは十分わかりましたか？
- Java既にわかってる人はどの程度？

この講義の目指すところ

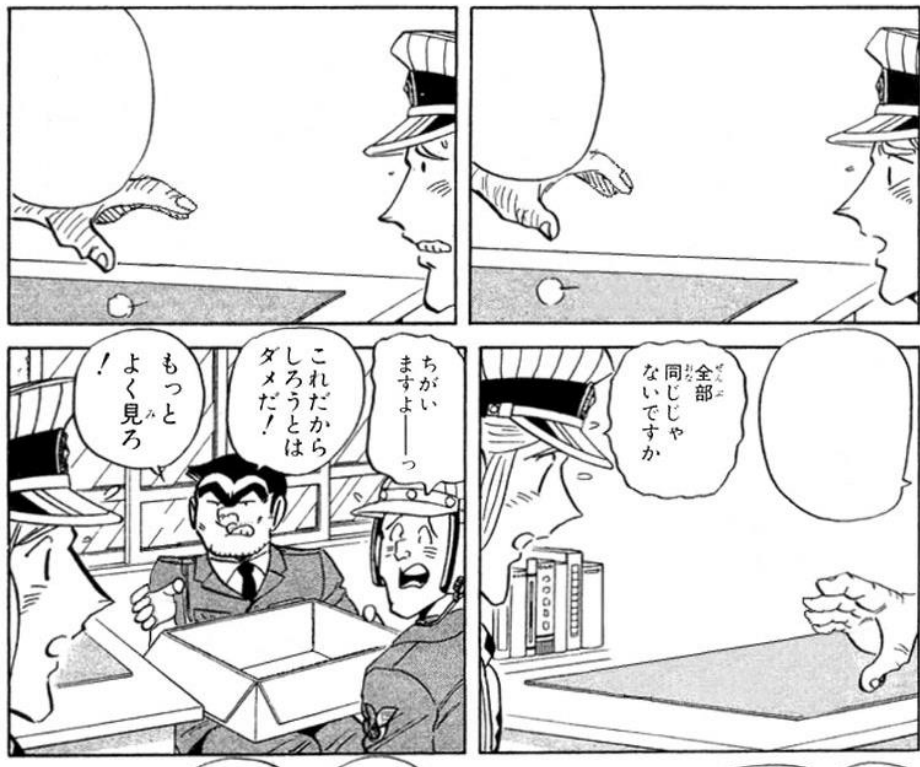
- 「新しい言語？多分書けると思いますよ。3日ください」
 - と言えるようになろう
 - 皆さんが今後しばしば出会うシチュエーション

Amazing Grace Hopper

- 3つぐらい言語を学ぶとわかってくる



実は簡単



- 考え方がわかれば同じように見えてくる
 - 要するにこれでしょ、と言えれば一人前

言語より考え方を学ぼう

- データを固まりにして扱うとか
- なるべく他人に影響されない作り方をしようとか

- 機能

- プログラミング・パラダイム

などを学んでいこう

そのための具体例としてJavaを学ぼう

皆様ご存じChatGPT

- かなり使えるやつ


DA

曜日の英単語を配列に格納し、順番にその頭3文字を表示するJavaコードを書いてください




以下は、曜日の英単語を配列に格納し、順番にその頭3文字を表示する Javaのコード例です。  

java

 Copy code

```
public class Main {  
    public static void main(String[] args) {  
        // 曜日の英単語を配列に格納する  
        String[] daysOfWeek = {"Sunday", "Monday", "Tuesday", "Wed"  
  
        // 順番にループして、各曜日の頭3文字を表示する  
        for (int i = 0; i < daysOfWeek.length; i++) {  
            String day = daysOfWeek[i];  
            String firstThreeChars = day.substring(0, 3);  
            System.out.println(firstThreeChars);  
        }  
    }  
}
```

 Regenerate response

プログラミングなんてもう必要ない？



プログラミングはもう少し広い

- ここで見えているのはかなりお膳立てされた部分
 - どんなプログラムを作ってほしいか、を割ときちんと定義してコードを書かせている
 - いわゆる「コーディング」
- 授業の課題みたいなものはわりと書けそう
- 書いてほしい「定義」の部分をきちんと書けるか？が人間に求められる
 - 問題の把握、設計 に当たる部分

DA

1から100までの数字で、3が付く数字だけ!を付けて順番に表示するJavaコードを書いてください



以下は、1から100までの数字で、3が付く数字だけ!を付けて順番に表示するJavaのコード例です。



java

Copy code

```
public class Main {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 100; i++) {  
            if (String.valueOf(i).contains("3")) {  
                System.out.print(i + "!");  
            } else {  
                System.out.print(i);  
            }  
  
            // 改行コードを出力する  
            System.out.println();  
        }  
    }  
}
```

これで良い、と自信をもって 言えるか？

- 「動きそう」で提出できる？
 - 成果物として責任をもって納品できる？ハンコ押せる？
 - 人間に残る仕事は「責任を取ること」
- 結局中身をチェックしないといけない
- チェックできる程度に理解してないとダメ
- ある意味、教員がやっているレポート採点みたいな能力が必要になるのかも

文句を付けられる？

- 動くけど、もっと良くできるね、と気づけるか
- さっきの例：
 - 「改行コードを後から付ける意味ある？出力に `println` 使えばよいのでは」と気づけるか？
 - 「確かに正しい判定ロジック。出力するなら無駄にならない。でも出力しないなら無駄かも」と気づけるか？
 - 実際、「3が付く数字を合計しろ」と言ったらコードが変わった。賢いね。

講義前・後の質問

- 横山まで
- dyokoyama@meiji.ac.jp

参考書

- スッキリわかるJava入門 第4版: 中山清喬 他, インプレス
 - 教科書として指定
- スッキリわかるJava入門 実践編 第4版: 中山清喬, インプレス
 - この講義の最後の方に関して、苦手な人向け
- Java本格入門 ～モダンスタイルによる基礎からオブジェクト指向・実用ライブラリまで～: 谷本心 他, 技術評論社
 - 得意な人向け
- 基礎からのJava（改訂版）: 宮本信二, SBクリエイティブ
 - 教科書として使う予定でしたが、品切れ増刷なしなので手に入りにくいかも
- どれか1セットくらい持っているとはかどる

本日の内容

- Javaの見た目、開発環境について知る
 - Eclipse
- 基本的な言語要素、制御構造を使えるようにする
 - main関数
 - 変数
 - if
 - ループ
 - 配列

Java とは

今のところは

- 比較的昔（**1996**）から長い間継続してメンテナンスされており
- 新しめの機能が入っており
- 実用性もある

言語、という程度の理解で結構です。

言語の特徴についてはまた後日講義します。

コンパイルするのが前提

- Javaのソースコードは .java
- コンパイルすると .class
- .classがJavaランタイムというプログラムと一緒に実行できる
 - JRE: Java Runtime Environment
- 参考: C言語はコンパイルするとLinuxやWindowsで直接動くバイナリが生成される
 - C, C++, Fortran,...
- スクリプト言語はソースコードをコンパイルせず、直接ランタイム上で動かす（ように見える）
 - Python, Ruby,...

Java ソースコードの構造

- まずはソースコードに「固まり」があることを意識しよう

ソースコード例

```
import java.util.Arrays;
import java.util.List;

public class HelloArray {
    int data;
    String stringdata;
    static int f(int x) { return x + 3; }
    public static void main(String[] args) {
        List<Integer> l = Arrays.asList(3, 6, 8);
        for (int x : l) {
            System.out.println("Hello: " + f(x));
        }
    }
}
```

まずは眺めてみる

- 「キーワードはCで見たのもあるな」
 - intとか、関数っぽいのか、for文とかがある
 - 文の最後は;で終わってるみたい
 - {}で部分部分がくくられてる
- 直感通り、文法はC言語に近い
 - int, double といった型や、if, for, whileなどの働きは同じ

大きな役割

このファイルに関する指示

```
import java.util.Arrays;  
import java.util.List;
```

```
public class HelloArray {  
    int data;  
    String stringdata;  
    static int f(int x) { return x + 3; }  
    public static void main(String[] args) {  
        List<Integer> l = Arrays.asList(3, 6, 8);  
        for (int x : l) {  
            System.out.println("Hello: " + f(x));  
        }  
    }  
}
```

クラス

大きな役割

このファイルに関する指示

```
import java.util.Arrays;  
import java.util.List;
```

クラス

```
public class HelloArray {  
    int data;  
    String stringdata;  
    static int f(int x) { return x + 3; }  
    public static void main(String[] args) {  
        List<Integer> l = Arrays.asList(3, 6, 8);  
        for (int x : l) {  
            System.out.println("Hello: " + f(x));  
        }  
    }  
}
```

データ (変数)

関数

C言語を思い出してみると

```
#include <stdio.h>

int data;
char* stringdata[];

int f(int x) { return x + 3; }

void main(int argc, char **argv) {
    int i;
    for (i = 0; i < 3; i++) {
        printf("Hello: %d¥n", f(i));
    }
}
```

C言語を思い出してみると

このファイルに関する指示

```
#include <stdio.h>
```

```
int data;
```

```
char* stringdata[];
```

データ (変数)

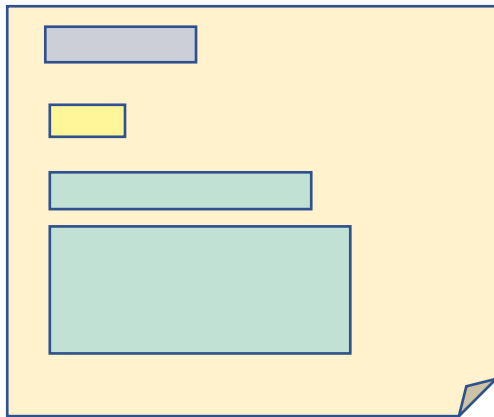
```
int f(int x) { return x + 3; }
```

```
void main(int argc, char **argv) {  
    int i;  
    for (i = 0; i < 3; i++) {  
        printf("Hello: %d¥n", f(i));  
    }  
}
```

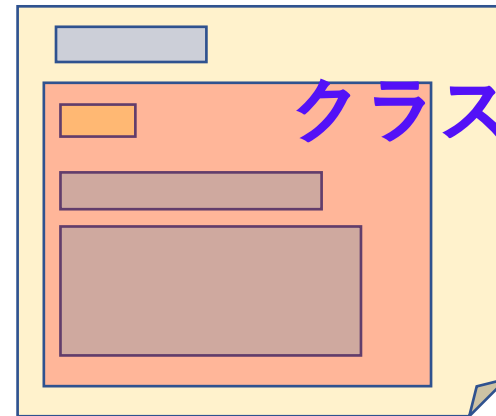
関数

C言語との違い

- class (クラス) というものがある
- クラスはオブジェクト指向で導入された大事なものの
- まずは「まとめり」だと思っておく
- Cではソースの中に部品がばらまかれていたのを、クラスでまとめている



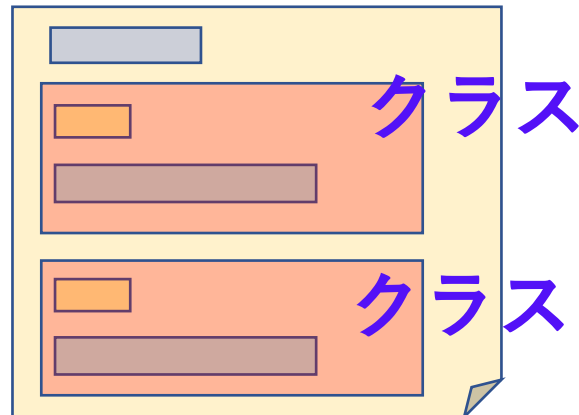
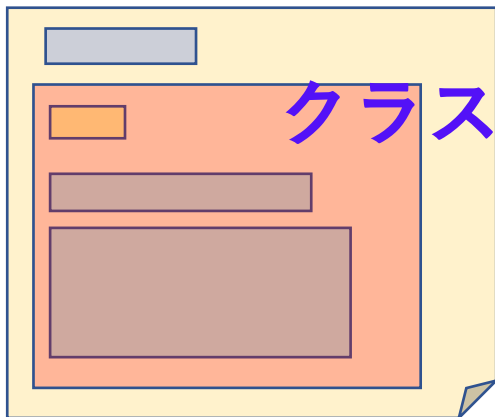
C言語



Java

Javaのクラス

- ソースにはクラスの定義しかない
 - クラスの外には何もない
 - 先頭にあるのはファイルに対する指示だけ
- 原則、1 ファイルには1 クラス
 - 複数のクラスを書くことも可能（そのうち説明する）



クラスの中

- 使える変数の宣言
- 関数（Javaではメソッド、と呼ぶ）
の2つしかない
- 変数宣言の形
 - 型の名前 変数名;
 - `int data;`とか
 - `int data = 3;`のように初期化をしている場合もある。
- 関数の形
 - 型の名前 関数名(引数,...) { 何かやること }
 - `int f(int x) { return x + 3; }`

練習

- Java入門っぽいキーワードでググったソースコード（超簡単なもの）を眺めてみよう
- 今説明した構造になっていることを理解しよう
 - どこがクラスの固まり？
 - クラスの中にあるものはちゃんと分類できる？
- ソースの中にまだ知らないキーワードはいっぱいあるはず
 - ひとまず無視して大きな構造を理解しよう

構造の理解

- 構造を理解してなくて混乱する人が非常に多い
- よくある流れ
 - 課題が出る
 - 「Java 配列 ソート」でググる
 - 出てきたコードを自分のソースの最後の方に張り付けてみる
 - 意味不明のエラーが出て動かない...
 - 変数名を自分のソースのものに合わせてみる
 - 反応ない...
 - }を付けてみる
 - ドツボ...。
- 出てきたコードを「どこに貼るべきか」がわからないとまずはダメ

読み書きの心構え(p.37)

- 構造に注意
 - 上から下へ、ではなく、外から内へ(p.42)
- インデントに注意
 - エディタがサポートしてくれるので、あえて崩さない

プログラムと実行

- プログラムの中に書いてあるのは「部品」だけ
- 部品が置いてある
 - メソッドが書いてあるが、置いてあるだけ
 - データも置いてあるだけ
- 書いたものが「実行」されるかどうかはわからない
 - 関数は呼ばれなければ実行されない
- 実行時には何らかの方法で「これを実行」と言わなくてはならない
- 普通は「実行のやり方」をあらかじめ決めてある
 - Cは「main関数を実行する」と決めてあった

main関数 (See p.46)

- 実行が始まるポイント
- この中に処理を書くと、コンパイル後にこの関数が最初に実行される、と決まっている

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

- クラスの大きな構造を意識しよう。どれがクラスの固まり？

手続きの部品

- 手続きの部品はただこれだけ：
 - メモリに名前を付ける：変数宣言
 - やりたい処理の流れを並べる
 - あらかじめ用意された演算を呼び出す
 - 足し算、代入、...
 - 関数を呼び出す
 - 自分で用意したものも、あらかじめ用意されたものもある
 - 処理の流れを変える
 - if文、for文...
- 関数の中に書いてある処理はこれらだけ
 - さっきのmain関数の中には何が書いてあるだろうか？

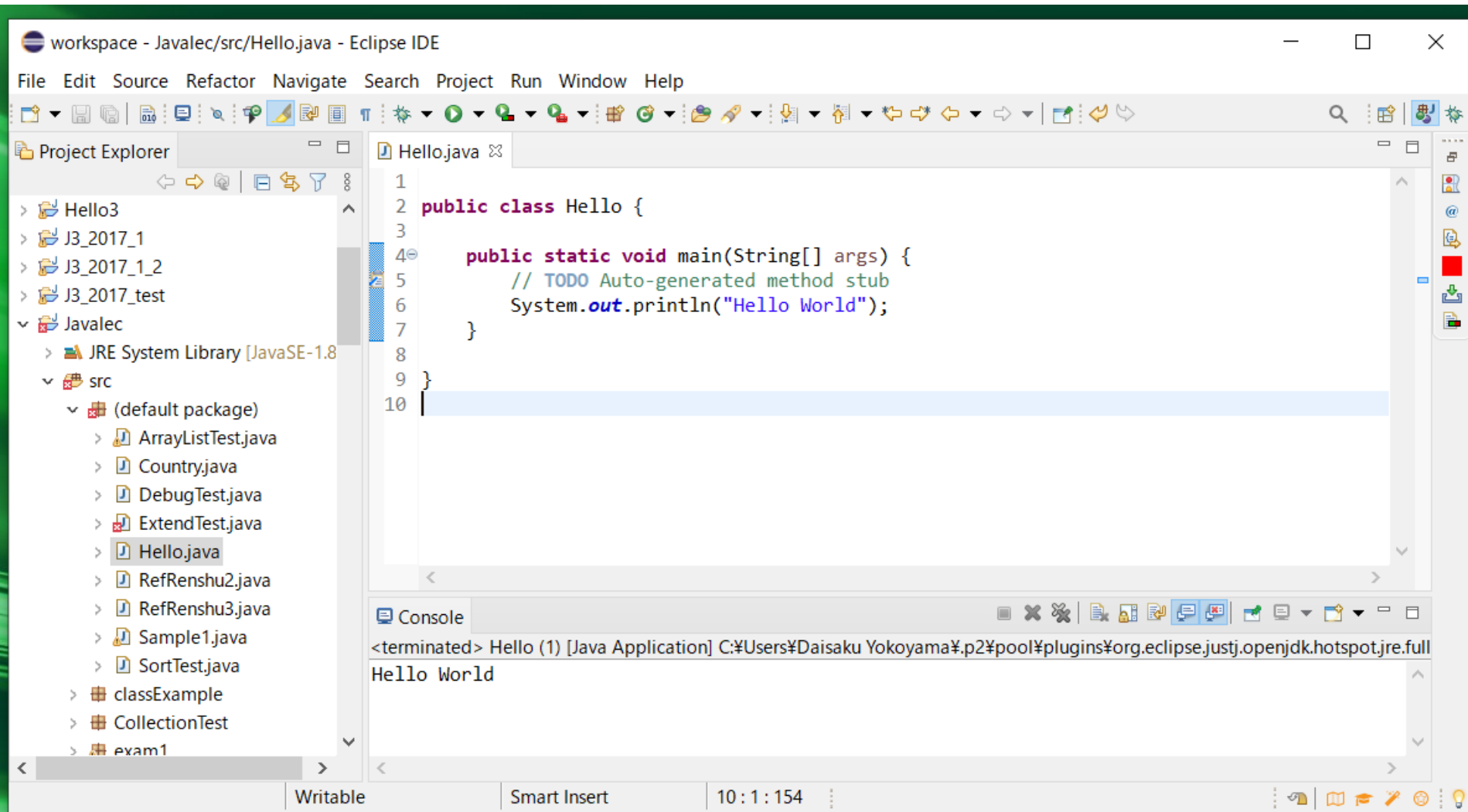

```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

- mainメソッドの中には「関数呼び出し」が書かれている
 - .でつながった長い名前になっているけど、今は関数名だと思っておくと、これは関数呼び出し
 - 関数呼び出しだから、引数には具体的な値が入っている
- mainメソッドそのものの引数と見比べてみよう
 - mainメソッドは「部品を定義している」
 - 引数(args)は実際に使うときに値を埋められて呼び出される
 - だから型の名前が付いてる

Eclipse

- Javaの統合開発環境
 - エディタ
 - サポート機能多数
 - ソースコード管理
 - コンパイラ
 - デバッガ
- Javaはコマンドラインで開発することも可能
 - See p.676～, 686～
 - Windows環境（not Linux）で説明してあるようです

Eclipse画面例



eclipseでのクラスの作り方

- Project Explorerでプロジェクト選択
 - 必要なら新規プロジェクト作成
- プロジェクト->src->package(とりあえずdefault packageでよい)で右クリック->New->Class
 - packageはソースファイルの固まり
- public static void mainを追加にチェック(optional)

```
public class Test {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
    }  
  
}
```

参考: package(とプロジェクト)

- Eclipseで開発を始めるときに聞かれたかも
- プロジェクト: アプリケーションとしてのまとめ、
くらいの意味
 - Eclipseの言葉 (Javaの言葉ではない)
 - 例えばJava演習の課題は全部1つのプロジェクトに入れて
おいてかまわない
- package: 複数のクラスをまとめて扱うための固まり
 - Javaの言葉
 - 例えば課題ごとにpackageを新しく作って提出、とすると
複数の課題が混ざらない
 - 今後の課題はそのような指定がある

アノテーション

- @Override なんて行が勝手に入ってた
- @XXXX という形式の「アノテーション」が用意されている
- 継承のところで説明します
- 今は無視（消しても良い）

コメント

- `/* Cスタイルのコメント(範囲指定) */`
- `// 1行だけのコメント`

```
public class Test {  
    public static void main(String[] args) {  
        // この行はコメント  
  
        /* これも  
           コメント */  
  
    }  
}
```

コンパイルと実行

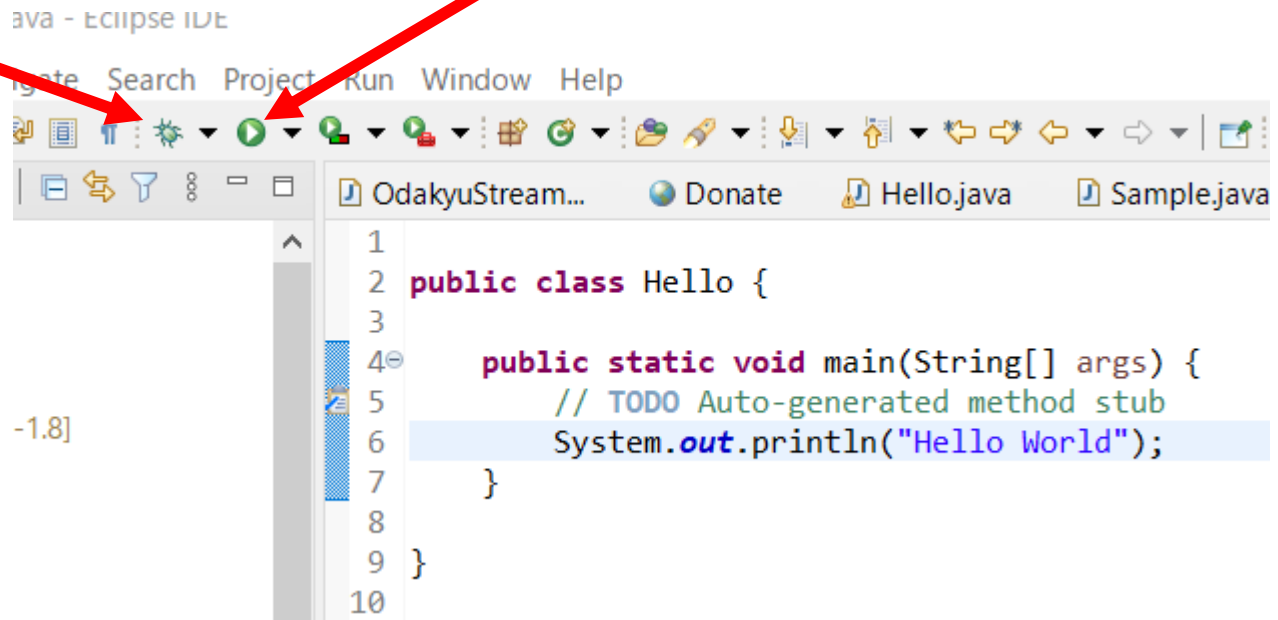
- Eclipseでセーブすると随時コンパイルが走る
- 実行は実行ボタンで
 - セーブできてないときに「コンパイルするか？」と聞かれることも
- コンパイルエラーがあると実行できない
 - 昔、エラーがなかった時に無事コンパイルした結果があると、そちらを実行していることもある
 - 「ソースを直しても結果が変わらない！」という誤解をよく聞く
- 行の横に赤い帯が出ているとエラーあり

Eclipseでの実行

- 実行してみよう

(参考)
デバッグボタン

実行ボタン



- 結果は下の“console”というウィンドウに出る
 - 標準出力

変数 (See p.48)

```
public class Hello {  
    public static void main(String[] args) {  
        int a;  
        a = 3;  
        System.out.println(a);  
    }  
}
```

- mainメソッドの中には何がある？

- 変数宣言
 - 代入（言語に用意された演算機能）
 - 関数呼び出し
- ですね

(用語)

- 宣言と定義
 - 関数宣言 と 関数定義
 - 変数宣言 と 変数定義
- 宣言は「こういう名前が今後出てくるからね」という予告
- 定義は「こういう名前の部品は中身これだからね」という実装
- 変数宣言は中身も作られる？
 - 今はそう見えるけど、今後の講義で違う場合も出てくるので楽しみに

変数 (See p.48)

- 名前
- 型 の 2 つが大事な属性
 - この 2 つを並べて書けば宣言できる
- 演算は型によって決まる
 - 文字列？ 数字？ 日付？
- 型が違うものは代入できない
 - 型が違うということは、プログラマーの意図しない状況になっているということ
- 変換できるときは暗黙のうちに変換されることもある
 - 便利のため

Javaに用意された変数の型 (See p.51)

- `int` 整数（普通の）
- `long` 整数（大きな整数）
- `double` 浮動小数点数（大きめの）
- `String` 文字列
 - C言語と大きく扱いが異なるので注意
- `boolean` 真偽値
 - これもC言語にはなかった
 - Cに慣れていると`boolean`が要求されるところに`int`を使って怒られがち

booleanが要求されるところ

```
public class Hello {  
    public static void main(String[] args) {  
        int a;  
        a = 3;  
        if (a) {  
            System.out.println("aは0以外");  
        }  
    }  
}
```

- if文は「真か偽か」で判断する部品
- Cは上のようを書けたが、Javaではダメ
 - if (a != 0) {...} と書けばOK。なぜか考えてみよう。

演算 (See p.74)

- 数値は普通の四則演算ができる
+, -, *, /, %(modulo)
- 真偽値に対する演算もある
&&, ||

if文 (See p.103, 115)

- 手続きは一本道ではない
- 複雑な条件はif {} else if {} else {}とか、論理演算子とか

for文 (See p.125～)

- 繰り返しがよく使われるので、専用の仕組みを作った
- よく出てくるので3種類も作った
 - for
 - while
 - do while
- ループの途中でちょっと特別処理をする仕組みも作った
 - continue
 - break

配列 (See p.142)

- まとめて処理する変数はまとめておきたい
 - 支出したお金一覧とか
- まとめたものを「配列」として、まとまったものに名前を付ける
 - `int[] a;` はまとまったものの全体に`a`という名前がつく
 - 1個1個には`a[0]`, `a[1]`,...という名前がつく
- 配列は作る時に “new” が必要

```
int[] a;  
a = new int[3];  
a[0] = 5;
```

拡張for または for each文 (See p.156)

- 「全データに処理をしたい」のであって、iを増やしたいわけじゃない、という場面は多い
 - 何番目の要素とかに興味があるわけじゃない
- 配列aに対し、
for (int x: a)
のように書くとa全体から1個ずつ要素を取ってくるループが書ける

提出課題1: ループと配列

- `javalec1`というパッケージを作り、`Looptest`クラスを作ってください
- `main()`メソッドの中に、`nums`という大きさ10の`int`の配列を準備してください
- `nums`に、3でも5でも割り切れない正整数を小さい数から（1から）順に10個入れてください
- `nums`の数を「0番目+9番目」「1番目+8番目」...のように、両端からペアにして足して、その値を1行に1個ずつ表示してください

(続き)

- 結果は、数字が5行出てくるものになるはずで
す
- 出力された結果と、組番号名前、をソースの頭
にコメントとして付けてください
- Looptest.javaのみをoh-meijiで提出してください
- ✂切: 4/16(火) 17:00

ヒント

- `int[] nums = new int[10];`
- 配列の中身を埋めるループと、配列の中身を取り出して表示するループ、に大きく分かれるでしょう
 - 中身を埋めるところはfor文よりwhileの方が書きやすいかも
 - 割ったあまりを求めるのは%でした
 - 表示のところはfor文でも良さそうですね
- 表示は`System.out.println();`にintの数字を与えます
- `javalec1`というパッケージを作ると、その中にあるすべてのソースコードの頭には
`package javalec1;`
という行を入れる必要があります。Eclipseでクラスの追加をすると勝手に入るかも。
- 配列の中身が埋まったところで一度表示して確認したほうが安心かも
 - 書くだけではなく、結果をきちんと確かめる癖をつけよう

本日のまとめ

- Javaの概要、開発環境について知る
 - Eclipseでの開発
- 基本的な言語要素、制御構造を使えるようにする
 - クラスという固まりが基本
 - main関数