

情報処理技法 (Javaプログラミング)1

第2回 コンピュータが情報を扱うには? (変数, データ型, 代入)

人間科学科コミュニケーション専攻
白銀 純子

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

第2回の内容

- ◆プログラムを書くときのお約束
- ◆エラーメッセージへの対処
- ◆プログラムで扱うデータのおはなし

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

出席確認

- ◆本日: 授業のページの「第2回授業の出席確認」にアクセスし、問題に解答
 - ◆授業のページ: <http://www.cis.twcu.ac.jp/~junko/Programming/>
- ◆次回以降は、授業開始前に、授業のページからアクセスして解答
 - ◆内容: 前回授業の復習問題
 - ◆タイムスタンプで、正規出席と遅刻を区別
 - ◆11:10までに解答したら正規出席
 - ◆トラブルで解答ができないときは申し出ること

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

前回の復習

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

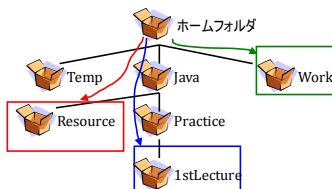
パス

- ◆パス: ファイルやフォルダのありかを表す文字の並び
 - ◆どのようにフォルダをたどれば、目的のファイルやフォルダにたどり着くかを表すもの
 - ◆「絶対パス」と「相対パス」に分類
 - ◆「A/B」*で、「A」というフォルダの中に「B」というフォルダまたはファイルが入っている、という意味
 - ◆Ex. 「Java/Practice/1stLecture」で、「Java」フォルダの中の「Practice」フォルダの中の「1stLecture」という意味
 - *Windowsでは「A\B」と表す
- ◆絶対パス: 最上位のフォルダから目的のファイルやフォルダへのパス
- ◆相対パス: 最上位以外のフォルダからのパス

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

相対パス

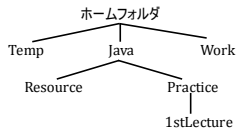
- ◆ここでは、ホームフォルダからのパスを考えてみる



Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

ホームフォルダからの相対パスの考え方(1)

- ホームフォルダから、目的のフォルダ・ファイルへの経路を「→」を使って書く
 - ◆ 経路: 「コンピュータ」から、どのようにフォルダをダブルクリックして開いていけば、目的のファイルやフォルダが見つかるか



Ex1. 「1stLecture」までの経路: **Java → Practice → 1stLecture**
 Ex2. 「Resource」までの経路: **Java → Resource**
 Ex3. 「Work」までの経路: **Work**

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

ホームフォルダからの相対パスの考え方(2)

- ◆ 「→」を「/」で置き換える

絶対パス

Ex1. 「1stLecture」までの経路:
 Java → Practice → 1stLecture ➡ **Java/Practice/1stLecture**

Ex2. 「Resource」までの経路:
 Java → Resource ➡ **Java/Resource**

Ex3. 「Work」までの経路: Work ➡ **Work**

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

「ターミナル」って何?

- ◆ ソフトウェアの名前(α)を入力することで、ソフトウェアを使うための道具
 - ◆ 普通、ソフトウェアを使うときには、そのソフトウェアのアイコンをダブルクリックすると、ソフトウェアが起動
 - ◆ ターミナルでは、ソフトウェアの名前(α)を入力し、「Return」キーを押すと、ソフトウェアが起動
- ◆ Javaプログラミング1で使うターミナル:
 - 「Finder」→「アプリケーション」→「ユーティリティ」フォルダを開き、その中の「ターミナル」をダブルクリック

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

コマンド入力の基本

- ◆ コマンドの形

「コマンド名」がソフトウェアの名前に相当

➡ **コマンド名 引数**

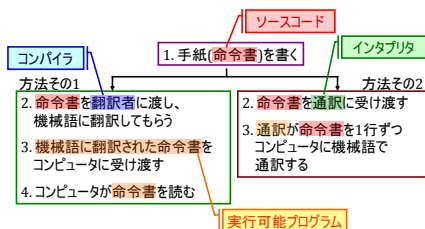
- 必ず「コマンド名」を最初に入力し、その後に「引数」を入力
- 「コマンド名」と「引数」の間にはスペースが1つ以上必要

例えば、コマンド名「ls」、引数「WWW」の場合:
 ➡ **ls WWW**と入力

- ◆ プロンプトは、「%」や「\$」と略して書かれることも
 - ➡ 「% abc」と書かれている場合には、「%」の後から入力すること
 (「%」は入力しない!)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

コンピュータとの会話のしかた(p. 16)



Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラミング言語とは(p. 17)

- ◆ コンピュータに命令を伝えるための言語
- ◆ 誰がいつ解釈しても意味が同じ
- ◆ 文法規則を厳密に定義

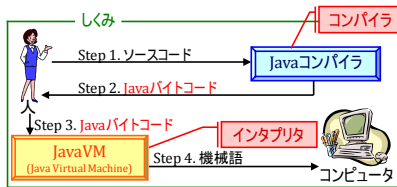
用語

手紙(命令書) ➡ **ソースコード(プログラム)**
 ソースコードを作成すること ➡ **プログラミング**
 ソースコードをコンパイルして機械語になったもの ➡ **実行可能プログラム**

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラミング言語Java(p. 26)

- ◆プログラミング言語の1つ
- ◆コンピュータやOSに依存せず、実行可能



Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

Javaプログラムの実行方法

- ◆Step 0: ターミナルの**カレントフォルダ**を、Javaファイルの保存場所に合わせる
 - ◆この作業は、コマンドプロンプトを起動したときに1度だけ行う
- ◆Step 1: Jeditなどでソースコードを作成する
 - ◆ファイル名は、必ず拡張子を「.java」とすること

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

Javaプログラムの実行方法

- ◆Step 2: ソースコードをコンパイルする
(コマンド名: **javac**, 引数: ソースコードのファイル名)
 % **javac** ファイル名 **.java**
 ➡ 「ファイル名.class」というファイルが作成される
- ◆Step 3: JavaバイトコードをJavaVMで実行する
(コマンド名: **java**, 引数: 拡張子なしのファイル名)
 % **java** ファイル名
 拡張子は不要

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムを書くときのお約束

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムの「カタチ」(p. 38)

- ◆プログラムには決まった形(最低限必要な命令のセット)がある
- ◆必ずこれだけの命令を書いてから他の部分を書く

```

JavaProg.java
import java.io.*;
import java.lang.*;

public class JavaProg {

    public static void main(String[] args) {

    }

}
    
```

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムの「カタチ」(p. 38)

```

JavaProg.java
import java.io.*;
import java.lang.*;

public class JavaProg {

    public static void main(String[] args) {

    }

}
    
```

プログラム中に書くお約束

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムの「カタチ」(p. 38)

```
JavaProg.java
import java.io.*;
import java.lang.*;

public class JavaProg {

    public static void main(String[] args) {

    }

}
```

import文

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

お約束「import文」とは(1)(p. 38)

- ◆ **import文**: Javaファイルの中で使われる機能(クラス)を明示する文
 - ◆ クラスの名前とパッケージをあわせて記述
- ◆ **パッケージ**: Javaに用意されている様々な機能(クラス)の分類
 - ◆ プログラミングのために使うことのできる機能(クラス)を、その内容ごとに分類したもの
 - ◆ 「java.io」、「java.lang」はパッケージの名前
 - ◆ 各クラスの名前は、「パッケージ名.名前」がフルネーム
 - ◆ 例えばファイルに対する操作をするための「File」というクラスは、フルネームは「java.io.File」
 - ◆ 「java.io.*」と書くと、「java.io」というパッケージに所属する全てのクラス、という意味

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

お約束「import文」とは(2)(p. 38)

- ◆ Javaファイルの中で使うクラスについては、そのクラスを使うことをファイルの冒頭で示しておくことが多い
 - ◆ 示した場合: そのクラスの名前をパッケージ名を省略可能(「java.io.File」であれば、「File」とだけ書けばよい)
 - ◆ 示しなかった場合: その機能の名前をフルネームで記述
- ◆ 示すには: 「import」というキーワードの後に続けてクラスの名前を書く
 - ◆ 1つ1つのクラスの名前を指定するか、「java.io.*」のように、1つのパッケージ内のクラスを全て、と指定する
 - ◆ 「*」で「全て」という意味

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムの「カタチ」(p. 40)

```
JavaProg.java
import java.io.*;
import java.lang.*;

public class JavaProg {

    public static void main(String[] args) {

    }

}
```

クラス宣言

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

お約束(1)「クラス宣言」とは(p. 40)

- ◆ Javaは、「クラス」というものを基本にして動作する
 - ◆ **クラス**: Javaプログラムを動作させるための基本単位
 - ◆ XXの処理をするためのクラス
 - ◆ XXのデータを格納するためのクラス
 - ◆ etc.
 - ◆ 様々な役割を持ったクラスをたくさん作り、お互いに連携させることでJavaのプログラムは動作
 - ◆ Javaプログラミング1の範囲では、クラスは1つか2つ
- ◆ 「public class クラス名 {...}」で、クラスの名前を決める

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムの「カタチ」(p. 40)

```
JavaProg.java
import java.io.*;
import java.lang.*;

public class JavaProg {

    public static void main(String[] args) {

    }

}
```

拡張子なしのファイル名

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

「拡張子なしのファイル名」とは(p. 40)

- ◆「public class クラス名 {...}」でクラスの名前を決める
 - ◆ Javaでは、原則として「クラス名」は、拡張子なし(「.java」なし)のファイル名にする
 - ◆ クラス名とファイル名は全く違うものにすることもできるが、原則として同じものにする
 - ◆ コンパイルすると、「クラス名.class」という名前のファイルができる
 - ◆ クラス名とファイル名(拡張子なしのファイル名)を全く違うものにしておくと、「ファイル名.class」というファイルはできない
 - ◆ プログラムを実行するときは、「java」コマンドの引数にクラス名を書く

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムの「カタチ」(p. 42)

```
JavaProg.java
import java.io.*;
import java.lang.*;

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

mainメソッド

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

「mainメソッド」とは(p. 42)

- ◆「この部分を最初に実行すること」という意味の命令
 - ◆ Javaでは、プログラムを実行したときに、まず最初に「public static void main(String[] args)」の「{ }」の間に書かれている命令を実行
 - ◆ 複数のクラスが存在するときは、「public static void main(String[] args)」があるのは1つのクラスのみ
 - ◆ 複数のクラスを使ってプログラムを実行するときは、「java」コマンドで指定するクラスは、「public static void main(String[] args)」を持っているクラス

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

ファイルの名前の付け方の注意(p. 40)

- ◆ ファイル名に使うよい文字(全て半角)
 - ◆ アルファベット(大文字・小文字ともOK、大文字・小文字は区別される)
 - ◆ 数字、「_」(アンダースコア)
- ◆ **ファイル名の1文字目は、数字にしないこと**
 - ◆ 1文字目はアルファベット又は「_」にすること
- ◆ 「int」や「double」などのJavaの中で定義されている言葉(予約語)をファイル名にしないこと
 - ◆ int, float, double, char
 - ◆ static, final, public, private, class, void, new
 - ◆ etc.

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

Javaファイルを保存するときの注意

- ◆ **日本語フォルダの中には保存しないこと**
 - ◆ ターミナルで日本語がうまく使えないため
- ◆ 保存するとき、Jeditの「漢字コード」の欄を「utf8」にしてから保存すること
 - ◆ MacでのJavaは、原則としてファイルの文字コードがUTF-8と考えている
 - ◆ 漢字コードの欄が「utf8」になっていないと、日本語を使ったときに文字化けする(コンパイルできないこともある)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムの書き方の基本

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムの処理内容の書き込み

- ◆ Javaプログラミング1では: mainメソッド内にすべての処理内容を書き込み

```
JavaProg.java
import java.io.*;
import java.lang.*;

public class JavaProg {

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

すべての処理内容を書き込み

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

処理の順番(p. 43)

- ◆ 書き込まれた命令は、上から順番に処理

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

    XXXX;
    YYYY;
    ZZZZ;

}
```

コンピュータは、書かれてある命令を上から順に実行
(1. XXXX
2. YYYY
3. ZZZZ
の順で実行)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

区切り文字(p. 44)

- ◆ 日本語を書くとき
 - ◆ 「、」で文節を区切る
 - ◆ 「。」で文を区切る
- ◆ 英語を書くとき
 - ◆ スペースまたは「、」で単語を区切る
 - ◆ 「、」で文を区切る

単語や文節、文を区切るための区切り文字がある

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

Javaの「区切り文字」は?(p. 44)

```
JavaProg.java
import java.io.*;
import java.lang.*;

public class JavaProg {

    public static void main(String[] args) {
        int apple, orange, banana, pine, strawberry;
    }
}
```

文の区切り: 「;」(セミコロン)

単語の区切り方その1: スペース

単語の区切り方その2: 「,」(コンマ)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

カッコの対応関係(p. 44)

- ◆ カッコは内側から閉じる

```
JavaProg.java
import java.io.*;
import java.lang.*;

public class JavaProg {

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

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

Javaプログラムを作るには?

1. テキストエディタ [jeditやEmacsなど] で処理内容を記述する
「import java.io.*; ~」の処理内容を、こういったテキストエディタで記述する
2. 記述した処理内容をファイルとして保存する
◆ ファイル名の拡張子を「.java」にする
3. 保存したJavaファイルをコンパイル・実行する

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

エラーメッセージへの対処法(p. 49)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

うまくいかないときの確認事項

- ◆ 必要な命令が、必要な場所に書かれてあるか?
 - ◆ 命令Aは命令Bよりも上に書いていないといけない
 - ◆ 命令Aはこの「{ ~ }」の中に書いていないといけない
 - ◆ etc.
- ◆ カッコの対応付けが間違っていないか?
 - ◆ 開くカッコと閉じるカッコの数は同じになっているか?
 - ◆ 正しい場所でカッコを閉じているか?
 - ◆ etc.

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラミングでのエラー(p. 49)

- ◆ プログラム作成時に、エラーでうまくいかないことも多い
 - ◆ コンパイル時に表示されるエラー: コンパイルエラー
 - ◆ スペルミスをした
 - ◆ カッコを開き忘れ・閉じ忘れた
 - ◆ 必要な場所に必要な命令を書いていなかった, etc.
 - ◆ コンパイル後、実行時のエラー: 例外
 - ◆ 数を0で割ろうとした
 - ◆ 使ってはならない番号を使おうとした(配列など), etc.
- プログラム中の文法間違い, という意味のエラー
- プログラムに文法間違いはないが、何らかのミスでそれ以上実行できない, という意味のエラー

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

エラーメッセージの表示(p. 49)

- ◆ エラーメッセージは、ターミナル上に表示される
 - ◆ コンパイル・実行後に、ターミナルに、予期しないメッセージが表示されていたら、それをよく読むこと
- ◆ コンパイルエラーは一度にたくさん表示されることがある
 - ◆ 何もメッセージが表示されず、プロンプトが戻ってきたときは、コンパイルが成功
 - ◆ メッセージが表示された場合は、コンパイルに失敗

④ コンパイルに失敗: プログラムの実行ができない!
➡ コンパイル→エラーの修正→コンパイル...
を繰り返してエラーをなくしてから実行!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

エラーに対処するために... (p. 49)

- ◆ Jeditの設定で、行番号と全角スペースを表示するようにしておく&便利
 - ◆ 行番号: メニューバーの「表示」→「行番号」→「パラグラフ」にチェック
 - ◆ 「パラグラフ」にしておかないと、エラーの行番号(ターミナルに表示される)とJeditでの行番号がずれる
 - ◆ 全角スペース: メニューバーの「表示」→「不可視文字の表示」→「全角スペース」にチェック
 - ◆ まちがえて全角スペースを書いてしまってもエラーになることもよくある

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

コンパイルエラーの基本形(p. 49)

基本的なコンパイルエラーのメッセージの形

XXX.java: n: メッセージ
プログラム中の文
^

XXX.java: コンパイルしたファイル名
n: エラーが見つかった行数(「n行目にエラーがある」という意味)
^: 「プログラム中の文」の中のアヤしい部分(間違っている部分)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

コンパイルエラーへの対処の基本(p. 49)

- ◆コンパイルエラーには一番上から順に対処すること
 - ◆コンパイルエラーがたくさん出てきたときは、多くの場合、上の方に出ているメッセージがより適切な意味
 - ◆1つのまちがいが影響していろいろな部分のメッセージを出すことも
 - ◆例えば、宣言していない変数を5箇所使っていたら、5つエラーメッセージが出てくる
- ◆「**メッセージ**」の部分をよく読み、エラーの意味を理解すること
- ◆Jeditで、エラーが出た**行番号**のところをよく見て、ミスを探すこと

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

例外の基本形(p. 51)

基本的な例外のメッセージの形

Exception in thread "main" **例外の内容**
at 例外の発生場所(**XXX.java:n**)

例外の内容: 発生した例外の意味(例外の名前)
XXX.java: 実行したファイル名
n: 例外が見つかった行数(「n行目に例外がある」という意味)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

例外への対処の基本(p. 51)

- ◆例外は、1度に1つだけしか表示されない(例外が出るとそこでプログラムの実行が終わってしまうため)
 - ◆何行もたくさん表示されることがあるが、発生した例外は1つだけ
 - ◆何行もメッセージが表示されたとしても、「**例外の内容**」を必ず確認すること
 - ◆何の例外が起こったのかを、きちんと理解すること
 - ◆例外が発生した行番号は、多くの場合、「at 例外の発生場所」の1つ目が適切
 - ◆1つ目の「at 例外の発生場所」を確認し、Jeditでその**行番号**の処理をよく確認して修正すること

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

エラー時のメッセージ

- ◆個々のメッセージは、教科書の各章の最後にあるので、よく見て対処していくこと
 - ◆各章の内容で、よく表示されそうなメッセージが掲載されている

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムで扱うデータ～種類～

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

スーパーマーケットの例(p. 56)

- ◆肉・魚系統
 - パックに詰める
 - ➡肉・魚の調理場へ
- ◆飲み物系統
 - ➡そのまま陳列棚へ
- ◆野菜・果物系統
 - 洗う・切る
 - ➡野菜・果物の調理場へ
 - コンピュータが扱うデータにも、様々な系統が存在
 - 届いた後、最初にどこへ持っていく?
 - ➡箱の中身を見なければわからない??

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

箱の種類で品物を分類(p. 56)

- ◆肉・魚
 - ダンボール箱
- ◆野菜・果物
 - 木箱
- ◆飲み物
 - プラスチック箱

分類された状態で届けば、どこに持っていけばいいかわかる

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

肉や野菜の種類は?(p. 56)

- ◆肉・魚
 - 牛肉, 鶏肉, 豚肉...
 - まぐろ, 鯛, さんま...
- ◆野菜・果物
 - キャベツ, きゅうり, にんじん...
 - りんご, みかん, バナナ...
- ◆飲み物
 - 牛乳, ジュース, お茶...

箱の表に種類を書いた紙を貼る
= 箱に名前を付ける

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムでは何を扱う?(p. 56)

- ◆スーパーマーケットでは「商品」を扱う
 - 品物の系統: 箱の種類で分類
 - 品物の種類: 箱に貼られてある紙で区別
- ◆プログラムでは?

扱うもの: データ

系統

主に整数, 小数, 文字

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムで扱うもの~データ型~(p. 56)

- ◆整数
 - プログラムでの表現: `int`
- ◆小数
 - プログラムでの表現: `float` または `double`
- ◆文字
 - プログラムでの表現: `char`

プログラムでのデータは、どの系統になるか決めておく必要あり

「データ型」と呼ぶ

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

データの「種類」は?(p. 57)

- ◆1つ1つのデータにそれぞれ名前をつける

例えば...

購入するりんごの数(int): `apple`

量った牛肉の分量(float): `meat`

自分が働いている陳列棚のエリア(char): `area`

「変数」と呼ぶ

系統(データ型): int
種類(変数名): apple

系統(データ型): float
種類(変数名): meat

系統(データ型): char
種類(変数名): area

「変数」= データを入れるための箱

データは原則として、必ず箱の中に入れて扱う

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムで扱うデータ~使い方~

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

変数の宣言(p. 59)

- ◆変数を使う(データを入れるなど)前に、変数を準備する必要

変数を「宣言する」という
= それぞれの箱が、「肉・魚系統」か「野菜系統」か「飲み物系統」かをコンピュータに知らせ、箱を準備する

```
int apple, orange, banana;
float meat, chicken;
char area, register;
```

スペース

準備(宣言)例

変数の系統(データ型)を先頭に書く

「,」で区切って複数の変数を予告(宣言)できる

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

変数の名前の付け方の注意(p. 60)

- ◆変数に使ってよい文字(全て半角)
 - ◆アルファベット(大文字・小文字ともOK、大文字・小文字は区別される)
 - ◆数字、「_」(アンダースコア)
- ◆変数名の1文字目は、数字にしないこと
 - ◆1文字目はアルファベット又は「_」にすること
- ◆「int」や「double」などのJavaの中で定義されている言葉を変数にしないこと
 - ◆int, float, double, char
 - ◆static, final, public, private, class, void, new
 - ◆etc.

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

変数の値(p. 62)

- ◆変数(箱)に値(データ)を入れて扱う
- ◆「=」で値を決める → 「代入する」という
= 箱の中に具体的なデータを入れること
- ◆用意した変数に初めて値を入れること: 初期化

購入したりんごの数が10個だった場合

apple = 10;

牛肉の分量を量ったら200.5gだった場合

meat = 200.5;

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

代入に使うデータ(p. 62)

- ◆代入をするとき、「=」の左側と右側は、同じデータ型でなければならない

つまり...「a = b」の場合

- aが「int」であれば、bも必ず「int」
- bが「float」であれば、aも必ず「float」

➡ aとbのデータ型が違う場合、コンパイルできない

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

変数の使い方(p. 65)

- ◆計算などの処理の際に、具体的なデータを書く代わりに変数名を書く

Ex. 100円のりんごを5つ買ったときの金額の計算
(りんごの個数の変数: apple)

100 × 5 の代わりに 100 × apple

実際の計算は、「apple」の中に入っているデータをコンピュータが取り出し、計算する

「参照する」と呼ぶ

変数を使うことのメリット

具体的なデータがいろいろと変わっても、プログラムの中を変更する必要がない
(Ex. りんごを5つ買った場合のプログラム、6つ買った場合のプログラム...というものを作る必要がない)

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

数の計算(p. 66)

- ◆変数(箱)の中には、計算結果や処理結果を入れることもできる

- ◆足し算: +
- ◆引き算: -
- ◆かけ算: *
- ◆割り算(商): /
- ◆割り算(余り): %

例えば...代金計算
(支払い金額: result)

100円のりんごを10個買った場合
apple = 10;
result = apple * 100;

100円のりんごを10個、
150円のバナナを5個買った場合
apple = 10;
banana = 5;
result = apple * 100 + banana * 150;

「result」には、「1750」という結果が入る

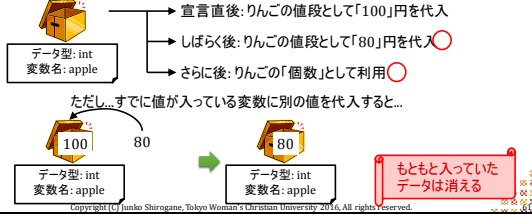
「result」には、「1000」という結果が入る

Copyright (C) Junko Shiragane, Tokyo Woman's Christian University 2016. All rights reserved.

変数の使いまわし

◆変数は使いまわし可能

- ◆同じ変数の宣言は1度だけで良く、何度も宣言する必要はなし
- ◆データ型の変更は不可



代入の不思議(p. 67)

「milk」を、店にある牛乳の在庫のバック数と考えと...
トラックが来る前: 在庫のバック数は30

milk = 30;

トラックが牛乳を50パック運んできた
この後の店の在庫数の計算は?



プログラム例(牛乳の在庫計算)(p. 67)

```
import java.io.*;
import java.lang.*;

public class JavaProg {
    public static void main(String[] args) {
        int milk;
        milk = 30;
        milk = milk + 50;
    }
}
```

牛乳の在庫数の宣言

現在の在庫数の代入

トラックが来た後の在庫数の計算

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

まとめ: 変数の鉄則

1. 扱うデータのデータ型を決定し、名前を付け、変数として宣言する
2. 宣言した変数を初期化する
3. 変数を参照して計算等の処理に使う

「宣言」→「初期化(値の代入)」→「参照」の順序を間違えないようにすること

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

変数宣言の注意(p. 61)

◆同じ名前の変数は、1回しか宣言できない

```
public static void main(String[] args) {
    int abc;
    .....
    int abc = 10;
}
```

コンパイルエラーが出る(一度宣言した変数は何回でも使えるので、「int abc = 10;」の「int」は不要)

◆変数には、宣言と同時に値を代入してよい

```
int abc;
abc = 10;
int abc = 10;
```

同じ意味を表す

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

変数の宣言忘れに注意(p. 61)

◆どの変数であっても、宣言していなければ使えない

```
int result;
result = banana + 10;
```

変数「banana」の宣言をしないうちに、「banana」のデータを使って計算しようとしている

「シンボルを処理解釈できません」というエラーメッセージ
宣言していない変数はすぐに書かれているので、よくメッセージを読んで宣言すること
※スベルミスの可能性もあるので、要注意

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

代入と参照の注意(p. 67)

◆初期化をしないと、変数を参照できないので注意

- ◆箱の中にデータが入っていないので、存在しないデータを使って計算などはできないため

```
int banana, result;  
result = banana + 10;
```

変数「banana」の初期化をしないまま、「banana」の中からデータを取り出して計算しようとしている

「変数**banana**は初期化されていない可能性があります」というエラーメッセージ

初期化が必要な変数(この変数を初期化すること)

※どのような値を代入すれば良いかはそのときでよく考えること

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

プログラムの記述とコンパイル・実行

◆プログラムの内容

- ◆JeditまたはEmacsなどのエディタ記述し、ファイルとして保存

◆コンパイル・実行

- ◆ターミナルで、保存したファイルを指定

エディタとターミナルは、どちらをどのように使うか、きちんと区別しよう!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

コンパイル・実行時の注意(1)

◆ターミナルでのカレントフォルダを、Javaファイルを保存しているフォルダに設定すること

- ◆カレントフォルダ: ターミナルでの、現在の作業フォルダ
- ◆ターミナルを起動したとき: カレントフォルダはホームフォルダ

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

コンパイル・実行時の注意(2)

◆カレントフォルダの変更のコマンド:

- ◆コマンドの入力
 - % cd **ホームフォルダからの相対パス**
- ◆Ex1. ホームフォルダの中で、「Desktop」→「Java」→「chap」に保存してある場合(相対パス: Desktop/Java/chap):
 - % cd **Desktop/Java/chap**
- ◆Ex2. ホームフォルダの中で、「Download」→「chap」→「chap01」に保存してある場合(相対パス: Download/chap/cha01):
 - % cd **Download/chap/cha01**

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

やってみよう!(2)

◆教科書p. 76の例題01-07をやってみよう

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.