情報処理技法
(Javaプログラミング)2

第6回
オブジェクト指向って?

人間科学科コミュニケーション専攻
白銀 純子

第6回の内容

- プログラミングの種類
- オブジェクト指向とは?

前回の出席課題の回答

- 「アルゴリズム」とは何か、下記のキーワードを使って説明しなさい。
  - キーワード: プログラム, 手順, 図

解答例

アルゴリズムとは、ある問題を解決するときの手順のことである。箇条書きの文章で書いたり、図を使って描くこともある。プログラムは、アルゴリズムをプログラミング言語で表現したものである。

プログラミングの種類

- 関数型言語
- 手続き型
- オブジェクト指向言語

関数型言語

- 数学的な関数のみをもとにして記述するプログラム言語
  - 一度変数に値が与えられれば、その変数の値は変化しない
  - 計算結果を引数とする、関数呼び出しのみで計算を行う
  - プログラミング言語: Lisp, Schemeなど

手続き型言語

- 記述された命令を上から順に実行していくプログラム言語
  - 処理の結果に応じて変数の値が変化
  - プログラミング言語: C言語, BASIC, Pascalなど

オブジェクト指向言語

- 「もの」と「もの」との関係に重点を置いて記述するプログラミング言語
  - ある「もの」に対して、それが持つ情報と、その「もの」が行う作業を記述する
  - ある「もの」と別の「もの」とのコミュニケーションを記述することで、プログラムを動作させる
  - プログラミング言語: SmallTalk, C++, C#など

Javaは?

- オブジェクト指向言語
- これまでの言語にはない、完全なオブジェクト指向を実現した言語
- 「Write Once, Run Anywhere」(一度記述すればどこでも動作する)がキャッチコピー
  - 一度記述すれば、OS等の環境が異なるコンピュータでもプログラムは動作する
  - 他のプログラミング言語では、OS等が違うとコンパイル・実行ができないこともある

オブジェクト指向の基礎

オブジェクト指向

- オブジェクト指向: 「もの」を中心してソフトウェアを構築する考え方
  - 「もの」: オブジェクト(インスタンスとも)
    - 1つ1つの具体的な実物
    - 名前を示されたとき、「これ」とそのものを特定できるもの
  - 「もの」の分類: クラス
    - 実物を分類したカテゴリ(実物の総称のような概念)
    - 名前を示されたとき、その概念にあてはまるものがいくつか存在するもの

※「オブジェクト」と「インスタンス」は厳密にはちがうもの

クラスって(1)?

- Javaは、「クラス」というものを基本にして動作
  - 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のデータを定義するためのクラス
    - etc.
  - それぞれの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

これまで書いてきたJavaプログラム

今回のクラス

クラスって(2)?

- 「public class クラス名 {」でクラスの名前を指定
  - 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

データを定義するためのクラス

- 異なる種類の情報をひとまとめにして扱うためのもの

一種のデータ型(ただし、「int」や「double」と違い、内部で色々な情報を持っている)

高校の生徒1人分の情報

```
String  name, address, tel;
int  studentNumber, english, math, language, science, society;
```

これだけの情報を持った「クラス」を作る

「クラス」は情報の集合体

- プログラムで扱うデータについて、何がひとまとまりかを定義したもの
  - 高校の生徒はどんな情報を持つ?
    - 住所, 氏名, 電話番号, 所属クラス, 試験の成績, etc.
  - 図書館の本はどんな情報を持つ?
    - タイトル, 著者, 出版社, 出版年, ID, etc.

「これでひとまとまり」と定義

ただし! 「クラス」は、具体的なデータは持たない
- 東京子さんの試験の成績は?
- 図書館の蔵書ID 0001の本のタイトルは?

具体的なデータを持つのは「オブジェクト」

「オブジェクト」って?

- 「クラス」とは、別の考え方をすれば、それぞれの人や物が「どういう種類の情報を持っているか」を表すもの

「Student」クラス(高校の生徒がどういう情報を持っているか)

出席番号1番の生徒(クラスA)
出席番号2番の生徒(クラスA)
.....
出席番号1番の生徒(クラスB)
出席番号2番の生徒(クラスB)

「住所」や「氏名」などの情報の持ち主
=オブジェクト

オブジェクト: 実際に具体的な情報を持っている人や物

クラスとオブジェクト(1)

- クラス
  - 実物を分類したカテゴリ(実物の総称のような概念)
  - 名前を示されたとき、その概念にあてはまるものがいくつか存在するもの

人や物を、持っている情報によって分類したもの
Ex. 東京女子大学の学生

- オブジェクト
  - 1つ1つの具体的な実物
  - 名前を示されたとき、「これ」とそのものを特定できるもの

「クラス」の分類に当てはまる、具体的な人や物
Ex. 東京女子大学の学生の東京子さん

クラスとオブジェクト(2)

図書館蔵書ID 0001:
児玉公信著: UMLモデリングの本質,
日経BP社

図書館蔵書ID 0002:
マーチン・ファウラー著, 羽生田栄一監訳:
UMLモデリングのエッセンス, 翔泳社

実物の本 =オブジェクト

「本」というカテゴリ(クラス)に分類

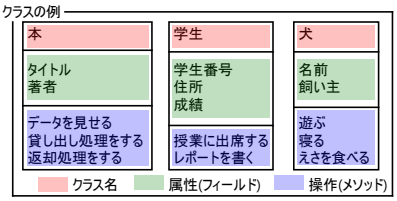
学生番号 k14x1001: 東京子
学生番号 k13x1001: 善福寺花子

実物の学生 =オブジェクト

「学生」というカテゴリ(クラス)に分類

クラスとオブジェクト(3)

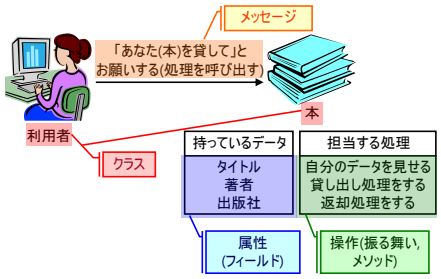
- クラス: 同じ属性と操作を持つオブジェクトの集合
  - 属性(フィールド):オブジェクトが持つ情報(データ)
  - 操作(振る舞い, メソッド): オブジェクトが担当する処理



クラスとオブジェクト(4)

- 1つのクラスにオブジェクトを所属させることができる
  - クラス: 実物を分類したカテゴリのようなもののため
- オブジェクト同士は、それぞれのクラスに定義された操作(処理)を呼び出す
  - 操作(処理)の呼び出しを「メッセージ」と呼ぶ
- メッセージを組み合わせてオブジェクト同士がコミュニケーションすることでプログラム全体が成り立つ

属性・操作・メッセージ(例)



プログラムでのクラスとオブジェクト

プログラムでしなければならないこと

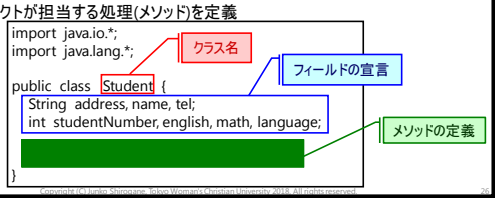
1. クラスを定義する
   - それぞれの「もの」について、内容を定義する
     - どのような名前か?
     - どのような情報(属性)を持っているか?
     - どのような操作(メソッド)を持っているか?
2. オブジェクトを作る
   - クラスに所属する個々のオブジェクトの情報の入れ物を作成
3. オブジェクトにデータを設定する
   - 2. で作ったオブジェクトに、具体的なデータを設定

原則

- データを定義するためのクラス(Javaファイル)を1つ作成
  - 処理のクラスとは別に作成
- 処理をするためのクラス(Javaファイル)を1つ作成
  - データ定義のクラスとは別に作成
- 処理のクラスの中で、データ定義のクラスのオブジェクトを作成
- 処理のクラスの中に、オブジェクトを使って、様々な処理を記述

1. クラスの定義のしかた

- これまでと同じ
  - 1ファイル1クラス
  - オブジェクトが持つデータ(フィールド)を変数として宣言
    - どのメソッドにも含まれない場所で宣言
  - オブジェクトが担当する処理(メソッド)を定義



「static」キーワード

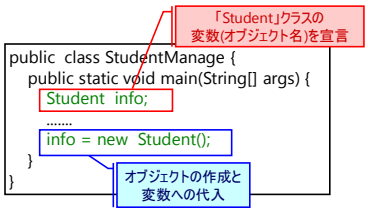
- フィールドの変数に「static」というキーワードをつけて宣言することがある
  - Ex1. static String schoolName;
  - Ex2. static int classNumber;
- staticなしのフィールド(インスタンス変数)
  - オブジェクトごとに値が異なるフィールドを表現するために利用
    - Ex. 1人1人の生徒の住所や電話番号、試験の成績など
- staticつきのフィールド(クラス変数)
  - どのオブジェクトも値が同じであるフィールドを表現するために利用
    - Ex. 学校の名前など

プログラムでしなければならないこと

1. クラスを定義する
   - それぞれの「もの」について、内容を定義する
     - どのような名前か?
     - どのような情報(属性)を持っているか?
     - どのような操作(メソッド)を持っているか?
2. オブジェクトを作る
   - クラスに所属する個々のオブジェクトの情報の入れ物を作成
3. オブジェクトにデータを設定する
   - 2. で作ったオブジェクトに、具体的なデータを設定

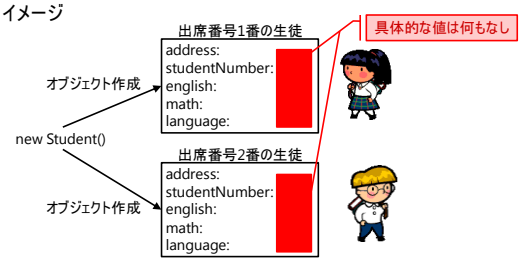
2. オブジェクトの作り方

- 「new クラス名()」でオブジェクトを作成し変数に代入
  - この作成・代入処理は、1. のクラスとは別のクラスのメソッド内で行う



「オブジェクトを作る」とは?

- 具体的な情報が何も設定されていない、情報の入れ物を作る、というイメージ



「オブジェクト」が複数ある場合

- 高校の生徒: 何人も存在

StudentManage.java

```
public class StudentManage {
    public static void main(String[] args) {
        Student info;
        .....
        info = new Student();
    }
}
```

これだと、1人分の情報だけ

オブジェクトを配列またはArrayListにする

複数のオブジェクトの扱い～配列～(1)

- オブジェクト: プログラムでの表記は変数と同じ
  = これまでのintやStringと同様に配列の宣言が可能

```
public class StudentManage {
    public static void main(String[] args) {
        Student[] info = new Student[50];
        .....
        info[0] = new Student();
        info[1] = new Student();
        .....
    }
}
```

「Student」クラスの
オブジェクトを50個分宣言

複数のオブジェクトの扱い～配列～(2)

- これまでと同様、「オブジェクト名[添え字] = new クラス名();」で作成
  - 配列で扱う個々のオブジェクトの作成を忘れないこと

```
public class StudentManage {
    public static void main(String[] args) {
        Student[] info = new Student[50];
        .....
        info[0] = new Student();
        info[1] = new Student();
        .....
    }
}
```

オブジェクトを1つ1つ作成(for文やwhile文でまとめて作成してもOK)

[ ]と( )の違いに注意!

- [ ]: 配列を表す
  - Student info[ ] = new Student**[30]**;
    ←変数「info」を、Studentクラスの30個の要素を持つ**配列として宣言**
  - Student info = new Student**( )**;
    ←変数「info」に、Studentクラスの変数として宣言し、**オブジェクトを代入**

「new Student...」と書いていても、意味が全く違うので注意!
- オブジェクトを配列にするときは、配列としての宣言と、各要素へのオブジェクトの代入が必要

```
Student[ ] info = new Student[30]; // infoを30個の要素を持つ配列として宣言
info[0] = new Student( ); // info[0]にオブジェクトを代入
info[1] = new Student( ); // info[1]にオブジェクトを代入
...
```


複数のオブジェクトの扱い～ArrayList～(1)

- オブジェクト: ArrayListで扱うことも可能

```
public class StudentManage {
    public static void main(String[] args) {
        ArrayList<Student> studentList = new ArrayList<Student>();
        .....
        Student info = new Student();
        studentList.add(info);
        .....
    }
}
```

「Student」クラスのオブジェクトを登録するためのArrayListの宣言

複数のオブジェクトの扱い～ArrayList～(2)

- 1つ1つオブジェクトを作成し、ArrayListに登録

```
public class StudentManage {
    public static void main(String[] args) {
        ArrayList<Student> studentList = new ArrayList<Student>();
        .....
        Student info = new Student();
        studentList.add(info);
        .....
    }
}
```

オブジェクトを1つ1つ作成(for文やwhile文でまとめて作成してもOK)

プログラムでしなければならないこと

1. クラスを定義する
   - それぞれの「もの」について、内容を定義する
     - どのような名前か?
     - どのような情報(属性)を持っているか?
     - どのような操作(メソッド)を持っているか?
2. オブジェクトを作る
   - クラスに所属する個々のオブジェクトの情報の入れ物を作成
3. オブジェクトにデータを設定する
   - 2. で作ったオブジェクトに、具体的なデータを設定

オブジェクトの利用(値の代入と参照)(1)

- オブジェクトの作成後、フィールドに値を代入可能
  - 「オブジェクト名.フィールド名」で普通の変数と同様に扱う
    - 「new」として、オブジェクトを作成したクラスのメソッド内で、「オブジェクト名.フィールド名」という変数を利用できる

```
public class StudentManage {
    public static void main(String[] args) {
        Student info;
        .....
        info = new Student();
        info.address="2-6-1, Suginamiku...";
        info.studentNumber=1;
        info.english=80;
    }
}
```

フィールドに値を代入

オブジェクトの利用(値の代入と参照)(2)

- 「オブジェクト名.フィールド名」で、「フィールド名」として使えるのは 1. で定義したクラスのフィールドの変数
  - 「オブジェクト名.フィールド名」で、「オブジェクト」「の(.)」「フィールド名」という意味

Student.java

```
public  class  Student  {
   String  address, name, tel;
   int  studentNumber, english, math, language;

}
```

StudentManage.java

```
public  class StudentManage {
   public static void main(String[] args) {
      Student  info;
      .....
      info = new  Student();
      info.address="2-6-1, Suginamiku...";
      info.studentNumber=1;
      info.english=80;
   }
}
```


オブジェクトの配列化～代入～(1)

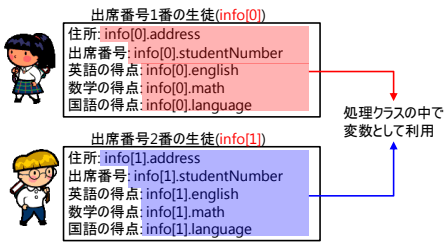
- 「オブジェクト名[添え字].フィールド名」で、通常の変数と同様に扱う

```
public  class StudentManage {
   public static void main(String[] args) {
      Student[]  info = new  Student[50];
      .....
      info[0] = new  Student();
      .....
      info[0].address="2-6-1, Suginamiku...";
      info[0].studentNumber=1;
      info[0].english=80;
      .....
   }
}
```

オブジェクトのフィールドに1つ1つ値を代入

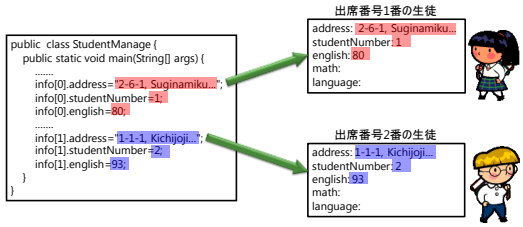
オブジェクトの配列化～代入～(2)

- 「配列の要素.フィールド名」で、個々のオブジェクトの情報を表現



オブジェクトの配列化～代入～(2)

- フィールドに値を入れることにより、各オブジェクトの固有のデータが設定

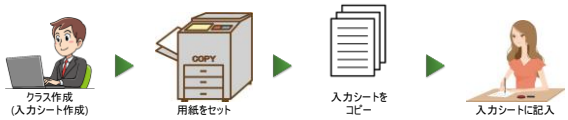


オブジェクトの配列化～利用～

- オブジェクトを配列にしたときも、添え字の考え方はこれまでと全く同じ
  - 添え字は0から数え始める
  - 0～[宣言した数-1]の番号の添え字を利用できる
    - ..., -3, -2, -1や、[宣言した数], [宣言した数+1], [宣言した数+2], ...は使えない
  - 高校の生徒などの場合、添え字と出席番号を対応させると扱いやすい
    - Ex. 出席番号1番の生徒は添え字0, 出席番号2番の生徒は添え字1, ...

クラスと配列のオブジェクトのイメージ

- クラス作成～フィールドへの値の代入は、個人情報の入力シートの作成をして、シートに入力するまでの流れのイメージ
  - クラス作成: 個人情報の入力シートの作成
  - オブジェクトの変数(配列)宣言: コピー機に用紙をセット
  - オブジェクトの作成(配列): コピー機で個人情報の入力シートをコピー
  - フィールドに値を代入: 1人1人がシートに記入



オブジェクトのArrayList化～代入～

- オブジェクトのフィールドに値を設定後、ArrayListに登録
  - ArrayListに登録後、フィールドに値を設定するのはややこしいので注意!

```
public  class StudentManage {
    public static void main(String[] args) {
        .......
        info.address="2-6-1 Zempukuji, Suginami-ku, ...";
        info.studentNumber=1;
        info.english=80;
        studentList.add(info);
        .......
    }
}
```

オブジェクトのフィールドに1つ1つ値を代入し、ArrayListに登録

オブジェクトのArrayList化～利用～

- 「get」や「size」などのメソッドはこれまでと同様に利用可能
- ArrayListならではのfor文の書き方も利用可能

```
int i;
Student st;
for (i = 0; i < studentList.size(); i = i + 1) {
    st = studentList.get(i);
    処理内容(「st.studentNumber」の形の変数も利用可能)
}
```

```
for (Student st: studentList) {
    処理内容(「st.studentNumber」の形の変数も利用可能)
}
```

同じ処理

NullPointerException

- オブジェクト指向プログラミングでよく見る例外
- オブジェクトを作成せずに、オブジェクトのフィールドを使おうとしているときの例外
  - コピー前の用紙(白紙)の入力欄を使おうとしているイメージ

```
public  class StudentManage {
    public static void main(String[] args) {
        Student[]  info = new  Student[50];

        info[0].address="2-6-1, Suginamiku...";
        info[0].studentNumber=1;
        info[0].english=80;
        .......
    }
}
```

info[0] = new Student(); が必要

コンパイルと実行

コンパイルと実行のしかた

- コンパイル
  - 「javac」の後に、ファイル名をスペースでつなげて複数のファイルをコンパイル

```
% javac  StudentManage.java  Student.java
```

  - または、「*」でそのフォルダに保存されているJavaファイルすべてをコンパイル
    - プログラムに関係ないJavaファイルもコンパイルされる。関係ないJavaファイルにコンパイルエラーがあれば、コンパイルが完了しないので注意

```
% javac  *.java
```

- 実行
  - 「java」の後に、「public static void main」が書かれているファイル名(拡張子なし)を書く

```
% java  StudentManage
```


やってみよう!(1)

- 高校の生徒5人分の名前・出席番号・5教科の得点の平均点を管理するクラスを作り、下記のように5人の情報を順番に表示するプログラム
  - 出席番号 名前 平均点
  - 1 東京子 80.3
  - 2 善福寺花子 83.4
  - .....
- 友達の名前とメールアドレスを管理するクラスを作り、標準入力から名前が入力されたらメールアドレスを表示するプログラム

やってみよう!(2)

- 下記の2つのクラスを持つプログラム
  - 1つ目のクラス: 生徒クラス
    - 名前と出席番号、5教科の試験の得点を入れるフィールドを持つ
  - 2つ目のクラス: 処理クラス
    - 生徒クラスのオブジェクトに5教科の試験の得点を設定する
    - 生徒の5教科の試験の平均点を計算する
- お菓子の名前と値段を入れるフィールドを持つお菓子クラスを作成し、標準入力で入力されたお菓子の名前と値段をフィールドの値として代入するプログラム
  - 条件
    - お菓子の情報は5つ分入力するようにし、配列でオブジェクトを扱うプログラムと、ArrayListでオブジェクトを扱うプログラムの両方を作ること(つまり、プログラムを2つ作ること)
    - 代入した結果を標準出力に出力すること