10月～11月パート

Java基礎編
(クラス設計)

https://www.silhouette-illust.com/keyword/%E3%83%87%E3%83%BC%E3%82%BF%E3%83%99%E3%83%BC%E3%82%B9
https://thiscalifornianlife.com/2018/12/18/java11-01/

学ぶ内容

・クラス設計

Java・DB編で必要となる知識となります。

前のパートの説明見ましたか…？

※IntelliJ IDEAのJDKのバージョンは、17以降にしてください。

※本パートであまり重要でないワードに関しては、最後にまとめて外部リンクを添えて説明とします。

クラス設計とは？(前回パートより)

クラス設計には、様々な方法がある。C言語では関数を定義した際に引数や戻り値を設定したと思う。
クラス設計では、クラス自体には引数や戻り値を設定することはできない。(厳密には少し異なる)
その代わり、クラス設計をする際にメソッドと呼ばれるC言語でいう関数を多数定義することができ、そのメソッドに引数や戻り値を設定することができる。

メインのクラスで任意のクラスをインスタンス化し、呼び出したクラスの中にあるメソッドをインスタンスを利用して呼び出しやりたい処理を行う。

メインクラス

```
import …

public class hogehoge{
  public static void main(String[] args){
                          インスタンス化
  ABC abc = new ABC();
  DEF def = new DEF();          インスタンス
  def.aisatsu();
  System.out.print(abc.keisan(2));
  }
}
                    メソッド呼び出し
```

ABCクラス

```
public class ABC{

  void print(float a){
    System.out.println(a);
  }

  int keisan(int b){
    int y = b*b;
    return y;
  }
}
```

DEFクラス

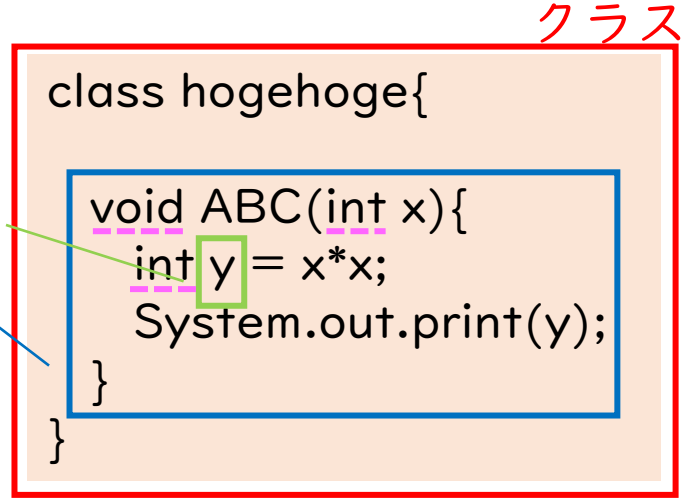
```
public class DEF{

  void aisatsu(void){
    System.out.println( “やあ” );
  }
}
```

メソッド

クラス設計①

★重要ワード

クラス　…　複数のメソッドを兼ね備えることができる大枠
メソッド　…　特定の機能を処理するためのプログラム
フィールド　…　クラス内でデータを保管するためのもの



クラスは、同一パッケージ内に複数(多数)作成することができる。
図ではhogehogeクラスを示しているが、hogehoge2,hogehoge3,…といったクラスを作ることも可能である。

メソッドは、ひとつのクラス内に複数(多数)作ることができる。例えば、四則演算ができるクラスを作りたい場合、各演算についてのメソッドを4つ作ればいいわけだ。ただし、引数と戻り値のデータ型を設定する必要がある。(表1)

フィールドは、データを格納する変数のようなものである。
フィールドにもデータ型を設定する必要がある。(表1)

例…　float x;　　floatがデータ型、xがフィールド

表1

型	説明	型	説明
int	32bitの整数型	char	16bitの文字型
long	64bitの〃	string	可変の文字列型
float	32bitの浮動小数型	boolean	1bitの論理型
double	64bitの〃		

※表1のデータ型は、主に使うものを抜粋したものであり、ほかにもある(byteとか)。

クラス設計②

★クラス設計の手順(まずは例を見て学ぼう)

①ex1クラス(メイン)の作成

```
public class ex1{
  public static void main(String[]
args){



  }
}
```

①

「psvm」と略した言い方をすることもある。

②任意の処理を記述

```
public class ex1{
  public static void main(String[]
args){
  int x1 = 5;
  int x2 = 2;
  int f = 2;

  int x3 = x1 + x1;
  int y = x3 ^ f;

  System.out.println(y);

  }
}
```

②

処理の記述は、psvm内に記述する必要がある。

クラス設計②

ex1クラスのように、すべての処理をメインクラスに記述するとどうなるのか…

```
   1  public class ex1{
   2    public static void main(String[]
   3  args){
   4
   5    int x1 = 5;
   6    int x2 = 2;
   7    int f = 2;

 …                        …

 999    System.out.println(y);
1000
1001
1002  }
1003 }
```

長文になってしまう(‘Д’)

このメインクラスで、加算する動作が複数回行われるとする。加算処理の記述を毎回「int ○ = □ + △」とすると記述量が増えるし、他人がそのソースコードをパッと見た時に何の処理をしているかを理解することが大変になる。

どうしたらいいの…？？

加算処理を行うためのクラスを作ればよい。(役割分担)
そうすれば、毎回加算処理を記述する必要はなくなり、新たなクラス内のメソッドを呼び出せば良くなる。

クラス設計③

★クラス設計の手順(新たなクラスの作成)　「引数どうしの加算処理ができるようにしたい」

①keisanクラス(new)の作成

```
public class keisan{



}
```

②メソッドの作成

```
public class keisan{
  int plus(int x){


  }
}
```

②

③メソッドの中身を記述

```
public class keisan{

  int plus(int x){
    int y = x + x;
    return y;


  }
}
```

③

メインクラスの値を使いたいので、plusメソッドの引数は「整数型」。引数の値を加算した結果を返したいので、戻り値は「整数型」。

戻り値のデータ型がint型であるため、returnが必要である。

クラス設計③

★クラス設計の手順(新たなクラスの作成)　「引数どうしの加算処理ができるようにしたい」

④ex1クラスの変更

```
public class ex1{
  public static void main(String[]
args){

   int x1 = 5;
   int x2 = 2;
   int f = 2;
                                    ④
   int x3 = x1 + x1;
   int y = x3 ^ f;

   System.out.println(y);

  }
}
```

```
public class ex1{
  public static void main(String[]
args){

   int x1 = 5;
   int x2 = 2;
   int f = 2;                       ④
   keisan ken = new keisan();
   int x3 = ken.plus(x1);
   int y = x3 ^ f;

   System.out.println(y);

  }
}
```

keisanクラスを使いたいので<u>インスタンス化</u>をする。<u>インスタンス</u>が「ken」であるから、この中にkeisanクラスの機能が集約されている。

keisanクラスの中にはplusメソッドがあり、ken.plus(フィールド);とすることでplusメソッドを使うことができる。

問題演習

基礎問題(1)(頭の中で考えてください)

ソースコードを動かすために用意するクラスとして正しいのはどれか。

```
public class basic1{
    (1)    (String[]
args){

   int x1 = 5;
   int x2 = 2;
   int f = 2;

   int x3 = x1 + x1;
   int y = x3 ^ f;

   System.out.println(y);

  }
}
```

① public static int main

② private static void main

③ public static void main

④ private static float main

(1) [ ]

問題演習

基礎問題(2)(頭の中で考えてください)

減算機能があるminusメソッドを持ったKeisanクラスが別であるとする。空欄に当てはまるコードを考えなさい。

```
public class basic2{
  (1)
{

   int x1 = 3;
   int x2 = 3;

   (2);
   int y = ken.minus(x1,x2);

   System.out.println(y);

  }
}
```

※(2)のヒント

メインクラスとは別のクラスがあって、そのクラスを使いたい時には何をする必要があるのか…

(1)

(2)

補足

クラスにメソッドを定義する際、引数は多数用意することができる。

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

    int x1 = 4;
    int x2 = 9;

    keisan ken = new keisan();
    int y = ken.multi(x1,x2);

    System.out.println(y);

  }
}
```

```
public class keisan{

    int multi(int x1, int x2){
      int x3 = x1 * x2;
      return x3;
    }
}
```

例だと「int」型と「int」型だが、「int」型と「float」型とすることも可能

豆知識

public … アクセス修飾子という。そのほかにprivate,protectedといったものがある。アクセス修飾子は、クラス等がどこからアクセス可能であるかを示している。publicは、全てのクラスからアクセス可能。

static … static修飾子という。「静的な」という意味。クラスをインスタンス化しなくてもアクセス可能。

String[] args … コマンドライン引数という。実行時に、プログラムに値を渡すことができる引数。(詳細は割愛)

※つまりpsvmは、コマンドライン引数を持ち戻り値を必要とせず、全てのクラスからアクセス可能なインスタンス化が不要のプログラム本文を打ち込むクラスといえる。(イメージね)

補足

この後の練習問題で必要となる知識なので、ここで軽く覚えましょう。
(その他の文法については「Javaプログラミング」授業や外部サイトで学びましょう)

for文

```
for(初期化式；条件式；変化式){

    //繰り返し処理を行う内容

}
```

サンプルコード

```
for(int i=0 ; i < 10 ; i++){

    int x += i;

}
```

for

初期化式 … どんなフィールド値でループするのかを記述。
また、初期状態の値を定義。
条件式 … ループする条件を初期化式で定義したフィールド値を用いて記述。
変化式 … 初期化式で定義したフィールド値がどう変化するのかを記述。

if文

```
if(条件式){
    //条件式がtrue(つまり正しい)
      のときに実行される処理
}
else if(条件式2){
    //条件式2がtrueのときの処理
}
                ...
else{
   //条件式がfalseの時の処理
}
```

サンプルコード

```
int x = 100;

if(x > 100){
  System.out.println( “秀” );
}
else if(x > 50){
  System.out.println( “優” );
else{
  System.out.println( “不可” );
}
```

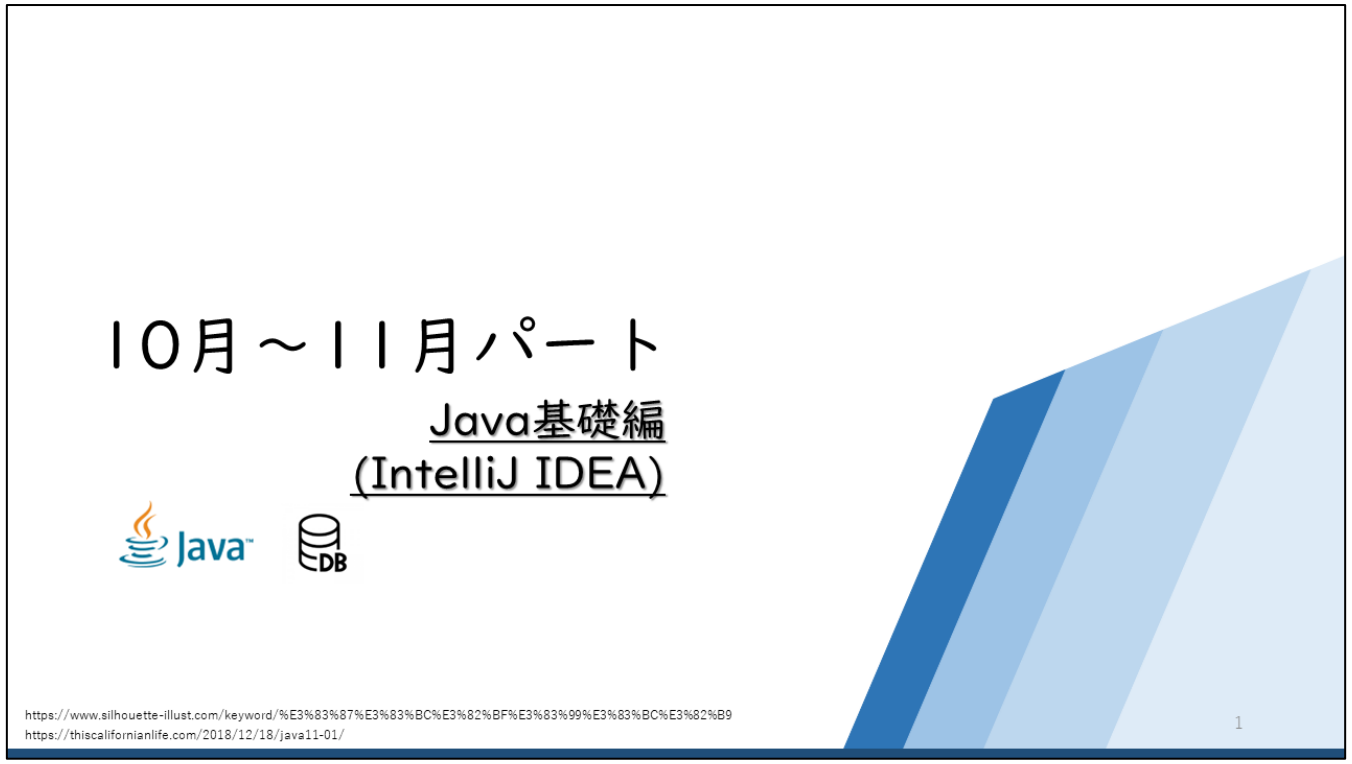
if

条件式 … 真偽を判断する条件を記述。

※他に問題を解くうえで必要な文法は、随時説明します。

問題演習の前に…

「IntelliJ IDEAでプログラミングをする」という資料を見てください。
この資料では、IntelliJでJavaプログラミングをする方法が書かれています。
それに沿って以降の問題演習を解いてください。



問題演習

練習問題1 (実際にIntelliJを使って実践しましょう)

　以下のメインクラスがある。このクラスでは、学籍番号を事前に登録している。任意で学籍番号を入力した時、その入力した学籍番号が事前に登録しているものと一致したら「complete!!」、不一致だと「error!!」と表示できるようになっている。
　学籍番号の判別からメッセージ表示までを別のクラスで処理したい。次の条件のもと、新しいクラスを作って同じ処理ができるようにしなさい。

```
import java.util.Scanner ; //Scannerライブラリを使うためのimport

public class prac1{
  public static void main(String[] args){

    String number = “B2001000” ;

    Scanner scan = new Scanner(System.in); //標準入力をするインスタンス
    System.out.print(“学籍番号を入力して下さい。:”);
    String gakuseki = scan.next();

    if(number.equals(gakuseki)){
        System.out.println(“complete!!”);
    }else{
        System.out.println(“error!!”);
    }

  }
}
```

～条件～

1. 新たなクラス名は「Gakuseki」
2. アクセス修飾子は「public」
3. メソッドは1つ(名前は自由)
4. メソッドの引数・戻り値は問題文・条件から考えること

※メインクラスもどこか変更するよ！！

問題演習

練習問題2 (実際にIntelliJを使って実践しましょう)

　次のページのようなメインクラスがある。このクラスでは、任意で5つの数値を入力し、その5つの数値をすべて加算した結果が49以下であれば「small」、50以上であれば「big」、100以上であれば「great!!」と表示するようになっている。
　メインクラスのソースコードが長くなっているため、新たなクラスを考えたい。以下の条件を満たすようなクラスを作り、元のメインクラスと同じ挙動をするプログラムを考えなさい。

～条件～

1. 新たなクラス名は「Keisan」
2. アクセス修飾子は「public」
3. 入力した数値５つの合計値を求めるメソッドと、合計値の判別をするメソッドの2つのメソッドを考えること(メソッド名は自由)
4. メソッドの引数・戻り値は問題文・条件から考えること

※メインクラスもどこか変更するよ！！

問題演習

練習問題2 (実際にIntelliJを使って実践しましょう)

```
import java.util.Scanner ; //Scannerライブラリを使うためのimport

public class prac2{
  public static void main(String[] args){

    num = new int[5]; //配列の定義(C言語とほぼ同じ考え方)

    Scanner scan = new Scanner(System.in);  //標準入力をするインスタンス
    for(int i=0;i<5;i++){
      System.out.print(“数字 %d つ目”,i);
      num[i] = scan.nextInt();
      System.out.println(“”); //改行
    }

    for(int j=0;j<5;j++){
      int x += num[i];
    }

    System.out.println(“合計値：%d”,x);
    if(x >= 100){
      System.out.println(“great!!”);
    }else if(x >= 50){
      System.out.println(“big”);
    }else{
      System.out.println(“small”);
    }

  }
}
```

ヒント

仮にGreetクラスがあり、インスタンス名はgreetとする。その中にはmorningメソッドとhelloメソッドがある。メインクラス内でmorningメソッド、helloメソッドの順で呼び出したいとき、

```
greet.morning();
greet.hello();
```

とすれば順番にメソッドを呼び出せる。

問題演習

練習問題3 (実際にIntelliJを使って実践しましょう)(少し厄介)

以下のようなプログラムを考えてもらう。特に縛りはないが、条件を満たすようにプログラムを考えてください。これといった正解はないので、頭を柔らかくしてプログラミングしましょう。

条件は、次のページに書いてある。

※様々な解釈の仕方があると思うので、自分で解釈したものをプログラミングしてください。

「事前に5つ以上の学籍番号とそれに対応するパスワードを定義する。学籍番号を入力してこの学籍番号が5つ以上の学籍番号に含まれていたらパスワードを入力できるようにする。入力した学籍番号が定義した学籍番号に含まれていなかったら「error!!」と表示し動作を終了するようにする。パスワードを入力して、入力した学籍番号と対応するパスワードと一致したら「ログイン完了」とメッセージを表示する。パスワードが一致しなかったら「不正なアクセス」と表示し動作を終了するようにする。」

問題演習

練習問題3 (実際にIntelliJを使って実践しましょう)(少し厄介)

～条件～

1. メインクラス1つ、メインクラスとは別のクラスを1つ以上作ること
2. メイン以外のクラスには、必ず1つ以上のメソッドを定義すること。
3. 全クラスのアクセス修飾子は「public」
4. 今まで習得した知識以外の文法を使ってもよい
5. 右の文法は必ず使うこと
6. 学籍番号とパスワードは、セットになるように二次元配列を使うこと

1．二次元配列
https://www.javadrive.jp/start/array/index7.html

2．強制終了(System.exit(0);)
https://www.sejuku.net/blog/52096

添付してあるリンクを見て使い方を覚えてください。

問題演習(解答)

基礎問題(1)(頭の中で考えてください)

ソースコードを動かすために用意するクラスとして正しいのはどれか。

```
public class basic1{
    (1)    (String[]
args){

   int x1 = 5;
   int x2 = 2;
   int f = 2;

   int x3 = x1 + x1;
   int y = x3 ^ f;

   System.out.println(y);

  }
}
```

① public static int main

② private static void main

③ public static void main

④ private static float main

(1) ③

問題演習(解答)

基礎問題(2)(頭の中で考えてください)

減算機能があるminusメソッドを持ったKeisanクラスが別であるとする。空欄に当てはまるコードを考えなさい。

```
public class basic2{
  (1)
{
    int x1 = 3;
    int x2 = 3;

    (2);
    int y = ken.minus(x1,x2);
    System.out.println(y);

  }
}
```

このソースコードを動作させるためには…??

なぜこれがあるのか…??

※(2)のヒント

メインクラスとは別のクラスがあって、そのクラスを使いたい時には何をする必要があるのか…

(1) public static void main(String[] args)

(2) Keisan ken = new Keisan()

その他出てきたワード

①パッケージ	https://kanda-it-school-kensyu.com/java-basic-contents/jb_ch07/jb_0702/
②ソースコード	https://blog.senseshare.jp/source-code.html
③アクセス修飾子	https://java-code.jp/134
④static修飾子	https://java-code.jp/130
⑤コマンドライン引数	https://tech.pjin.jp/blog/2021/07/29/java_07_01_command_line_argument/
⑥import	https://www.javadrive.jp/start/const/index9.html
⑦配列	https://www.sejuku.net/blog/14461

お疲れさまでした

次回は、インスタンス化実践です。