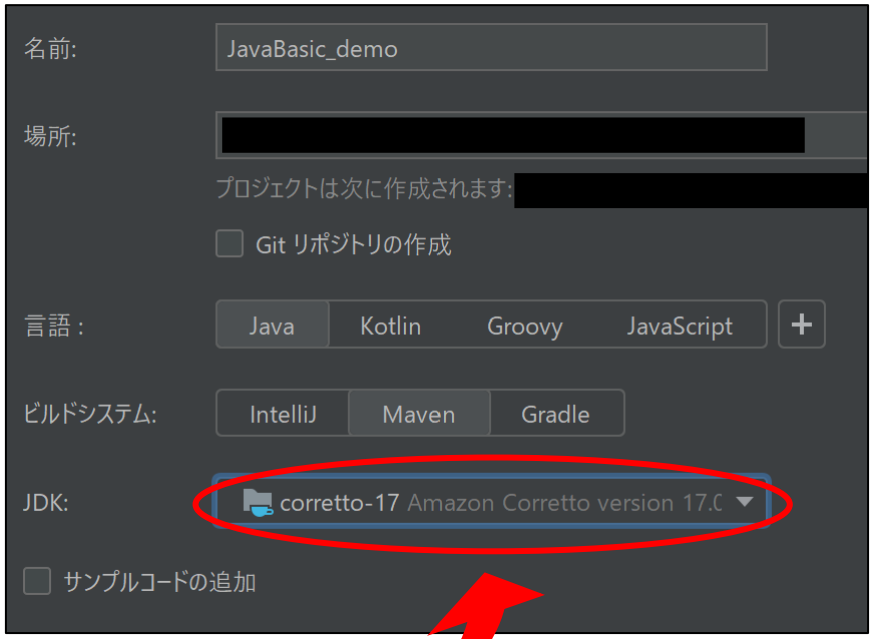
10月～11月パート

Java基礎編
(インスタンス化)

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

・インスタンス化



Java・DB編で必要となる知識となります。

前のパートのクラス設計見ましたか…？

※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。

インスタンス化とは？(前々回パートより)

インスタンス化とは、作成した新しいクラスを利用するための動作である。インスタンス化したクラスのメソッドの機能を集約したものをインスタンスという。特定のメソッドを呼び出したいときにインスタンスを利用してメソッドを呼び出す。(詳細は次次回パート)

```
import …

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

abc.○○()とすることで、ABCクラスの○○メソッドを呼び出している。

ABCクラスには2つのメソッドがあり、abcというインスタンスには、printメソッドとkeisanメソッドの機能がある

```
public class ABC{

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

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

メソッド

インスタンス化

★重要ワード

インスタンス化　…　インスタンスを生成するための動作
インスタンス　…　クラスをもとに生成された実体(オブジェクト)
メソッド　…　特定のクラス等に含まれるもの
コンストラクタ　…　あるクラスをインスタンス化した際に、そのクラスが行う処理(後に説明)

例：
```
ABC abc = new ABC();
abc.keisan(2);
```

```
import java.util.Scanner;

public class hogehoge{
  public static void main(String[] args){
                                  インスタンス化
   ABC abc = new ABC();
   Scanner scan = new Scanner(System.in);
   int x = scan.nextInt();
   System.out.print(abc.keisan(x));
                                  メソッド
  }
}
```

インスタンス

実はこれもインスタンス化。
「scan」がインスタンス

ABCクラスを使いたい場合、例の1行目のように宣言する。
「abc」の部分は自分がわかりやすいような単語で構わない。ABCクラス内にあるkeisanメソッドを呼び出したい場合は、例の2行目のように宣言する。

※メソッドの定義の仕方は、前回パートのクラス設計で学習済み。

コンストラクタ

★重要ワード

コンストラクタ　…　あるクラスをインスタンス化した際に、そのクラスが行う処理(後に説明)

今までは、メインクラス以外のクラスを定義する際、メソッドの定義しかしてこなかったが、クラスをインスタンス化した際の初期初期をすることができる。
メソッドと同様に引数を使うことができ、右図のようにコンストラクタを使う。インスタンス化の時は、以下の例のようにするとコンストラクタが行われる。

例　　ABC abc = new ABC(3,”Taro”);

※ this　：クラス内のフィールドを使いたいときに使うことがある。
super ：superを使いたいクラスとは別のクラスにあるフィールドを使いたいときに使うことがある。

```
public ABC {

  int number;
  String user_name;

  ABC(int num , String name){
    this.number      = num;
    this.user_name   = name;
  }

  void print(){

     System.out.println( “No. %d   Name %s”, number , user_name);
  }

}
```

※「this」というのは、「このクラス内の」という意味合いである。

コンストラクタ

コンストラクタの使う場面とは

コンストラクタといっても、前ページの説明だと別にメソッドで処理できるじゃんっていう話になります。このページでは、こういった場面でコンストラクタを使うと便利ということを例を見せながら紹介します。

右の例では、コンストラクタで累乗値mを設定し、funcsionメソッドでは、引数の2つの値を各自m乗し、その値を割った値をyに格納しyを返すといったKeisanクラスを定義している。

累乗の設定と計算過程を別々にすることで、人為的なミスを防ぐことに繋がる。メインクラスについては特に決めていないが、仮にメインクラスでこのKeisanクラスをインスタンス化し、funcsionメソッドを繰り返し処理するとする。この場合、ループ処理するたびに累乗を設定していては処理に負担がかかるし、Keisanクラスのインスタンス化時でコンストラクタをしておくと、funcsionメソッドで何かしらのエラーが起きても、累乗は別で定義しているため、大きな被害を受ける可能性が低くなる。

```
public Keisan {

  int m;
  float y;

  Keisan(int num){
    this.m  = num;
  }

  float funcsion(int x1 , int x2){
    y = x1^m / x2^m;
    return y;
  }

}
```

補足

前ページで、標準入力も**インスタンス化**であるといった。しかし、標準入力を使用する際、import宣言を使う必要がある。この標準入力の例だと、「import java.util.Scanner」と宣言する必要がある。その他にも様々なimport文があるが、「以降のその他出てきたワード」で他のimport文を紹介する。

import文を使う場合、図のように最初に宣言する必要がある。複数宣言可能。

今回は、標準入力を使うためのScannerをimportしたので、クラス名はScanner。コンストラクタがあるので、引数を指定。

前述でインスタンス名をscanとしたので、scanの中にあるnextIntというメソッドを呼び出す。

```
import java.util.Scanner;

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

    Scanner scan = new Scanner(System.in);
    System.out.print(“番号を入力して下さい。”);
    int num = scan.nextInt();
    System.out.println();

    System.out.println(“整理番号：%d”,num);
  }
}
```

問題演習

練習問題1 (実際にIntelliJを使って実践しましょう)(修正済み)

以下のソースコードは、クラス内で定義した名前と学籍番号を表示するプログラムとなっている。
任意で名前と学籍番号を標準入力できるようにしたい。
以下のソースコードに必要なコードを追加し、要件に合うようなプログラムをしなさい。Scannerクラスに含まれているメソッドはこのパートで紹介したnextInt以外にもあるので、各自調べなさい。

```
public class pra2_1{
  public static void main(String[] args){

    String name1 = “千歳太郎”;
    int num = 2001000;

    System.out.printf(“氏名：%s , 学籍番号：b%d”, name1 ,num);

  }
}
```

問題演習

練習問題2 (穴埋め問題です。IntelliJに穴埋めしたソースコードを移して実行してみましょう。)

以下の空欄に当てはまるコードを考え、IntelliJで実行結果を確認しなさい。その際、次のページの概要を見て考えなさい。

```
import java.util.Scanner;

public class pra2_2{
   public static void main(String[] args){

      String name_1 = “千歳花子”;
      int num_1 = 2110000;
      Scanner scan = new Scanner(System.in);
      Login log = new Login( ① , ② );

      System.out.print(“名前を入力して下さい。”);
      String name_2 = scan.next();
      System.out.print(“学籍番号を入力して下さい。”);
      int num_2 = scan.nextInt();
      String message = log.message( ③ , ④ );
   }
}
```

※次のページにLoginクラスのソースコードが書いてあります。

問題演習

練習問題2 (穴埋め問題です。IntelliJに穴埋めしたソースコードを移して実行してみましょう。)

このプログラムは、事前に名前と学籍番号が定義されている。任意で名前と学籍番号を入力し、事前に定義してあるものと一致したら「ログイン成功。」、不一致の場合は「ログイン失敗。」とメッセージが出るようになっている。

```
public Login(){
  String name;
  int num;

  Login(String name , int num){
    ⑤
    ⑥
  }

  void message(String name_n , int num_n){
    if(name.equals(name_n)){
      if(num == num_n){
        System.out.println(“ログイン成功。”);
      }
    }else{
      System.out.println(“ログイン失敗・”);
    }
  }

}
```

その他出てきたワード

①this https://qiita.com/takahirocook/items/d251ec4693c68f6b9538

②super https://camp.trainocate.co.jp/magazine/java-super/

③コンストラクタ https://www.javadrive.jp/start/constructor/index1.html

④import https://www.javadrive.jp/start/const/index9.html

⑤importの種類 https://docs.oracle.com/javase/jp/6/api/java/util/package-summary.html

おわり

次週は、Java&DB編です。