

オブジェクト指向 3

クラスとインスタンス (演習課題)

2017/11/6(月) プログラミングII 第五回
福井大学 工学研究科 情報・メディア工学専攻
長谷川達人

※WindowsでPC起動しておいてください.



本講義の概要

前半：Java 担当：長谷川		後半：Scala 担当：石井先生	
第1回	Java基礎文法/開発環境の使い方	第9回	Scala言語の基本
第2回	Java基礎文法/C言語との差異	第10回	関数型プログラミングの基本
第3回	オブジェクト指向	第11回	関数とクロージャ
第4回	クラス/インスタンス/メソッド1	第12回	関数型プログラミングの簡単な例
第5回	クラス/インスタンス/メソッド2	第13回	静的型付けと動的型付け
第6回	継承/カプセル化1	第14回	ケースクラス/パターンマッチング
第7回	継承/カプセル化2	第15回	多相性やパターンマッチングの例
第8回	中間試験/Java言語のまとめ	第16回	期末試験

※来年以降は全てJavaになる予定

本日の目標

概要

クラスの作り方, インスタンスの使い方がわかる.

目標

要求に従ったクラス設計, インスタンスの利用ができる.

本日の目次

- 復習と質問に対する回答
- 命名規則（前回の残り）
- 修飾子（前回の残り）
- 課題演習

復習の量がかなり多いので、
前回は余裕だった人は命名規則～
先読み＋演習課題をやって
いてくれてよい。



質問に対する回答

用語について

- 授業で出てきた用語はすべて覚えないといけないのか？
- 慣れないカタカタ用語が多くてつらい。
 - 回を重ねるにつれ覚えると思うが、次の用語程度は覚えてほしい。
- クラス : 設計図（構造体の宣言のようなもの）
- インスタンス : 設計図をもとに作成された実体
- メソッド : C言語でいう関数のこと
- フィールド : クラス内のグローバル変数的なもの
- コンストラクタ : インスタンスの初期動作を書く場所
- オーバーロード : 同じ名前のメソッドを定義すること
- コンパイル : ソースを実行できるように変換すること

クラスとインスタンス

復習

イメージとしては、

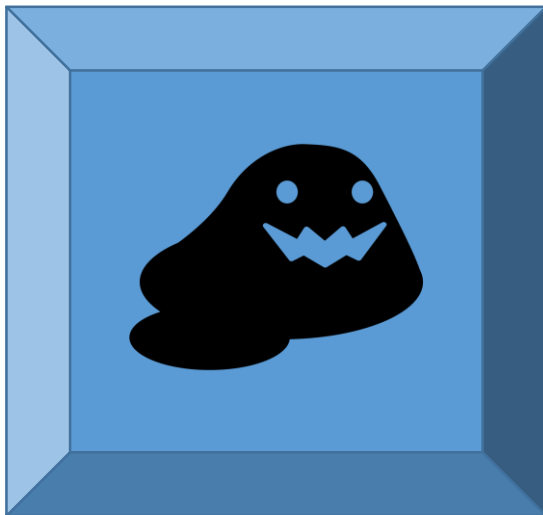
クラス

インスタンス

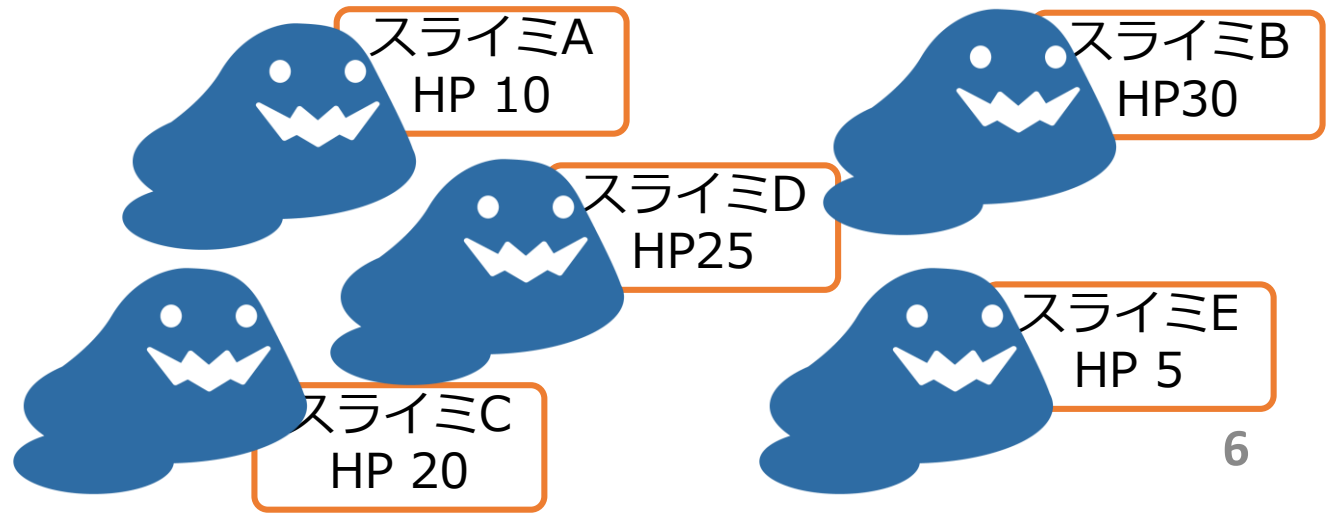
= 設計図や金型

= クラスをもとに作られた量産品

クラス（金型）



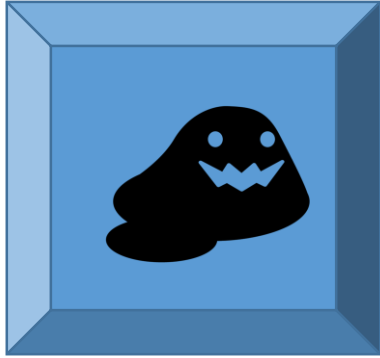
インスタンス（金型で量産）



クラスとインスタンス

復習

金型



Monster
クラスの定義

```
public class Monster{  
    int hp = 100;  
    String name = "";  
    public int attack(){  
        // 0~9までの乱数のダメージを返す  
        return (int)(Math.random()*10);  
    }  
}
```

```
public static void main(String[] args){  
    Monster m1 = new Monster();  
    m1.name = "スライミA"; // m1の名前を設定  
    m1.hp = 100;           // m1のHPを設定  
    Monster m2 = new Monster();  
    m2.name = "スライミB"; // m2の名前を設定  
    m2.hp = 150;          // m2のHPを設定  
}
```

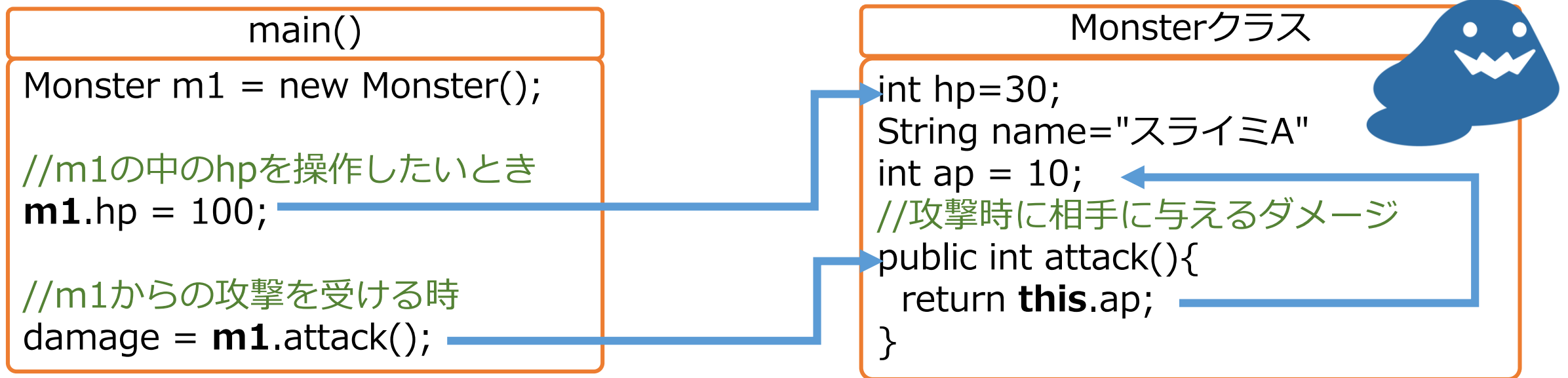
インスタンスの生成



質問に対する回答

this.について

- thisを使うタイミングがよくわからない.
 - m1.method() : m1内のメソッドorフィールドにアクセス
※m1はMonsterのインスタンスとする.
 - this.method() : 自クラスのメソッドorフィールドにアクセス



質問に対する回答

this.について

- Monsterクラスを少し簡単にして（乱数を使わず）考えてみる.
- 以下のプログラムでは, thisを使わなくても正常に動く.

```
public class Monster{  
    int hp = 100;           // ヒットポイント  
    String name = "スライミ";  
    int ap = 10;            // 攻撃力  
  
    // 誰かに攻撃するとき, 攻撃力分のダメージを与える  
    public int attack(){  
        return ap;  
    }  
}
```

apはフィールド
のapを指す

質問に対する回答

this.について

- 勿論, this.を使っても, 正常に動く.
- this.は自クラス内のフィールドを指し示すが, C言語のグローバル変数のようにそのままapと書くこともできるのである.

```
public class Monster{  
    int hp = 100;           // ヒットポイント  
    String name = "スライミ";  
    int ap = 10;           // 攻撃力  
  
    // 誰かに攻撃するとき, 攻撃力分のダメージを与える  
    public int attack(){  
        return this.ap;  
    }  
}
```

this.apもフィールド
のapを指す

質問に対する回答

this.について

- 一方、メソッド内でapという同名のローカル変数を使ってしまった場合にトラブルが起こる.

```
public class Monster{  
    int hp = 100;           // ヒットポイント  
    String name = "スライミ";  
    int ap = 10;            // 攻撃力  
  
    public int attack(){  
        int ap = 10;        // 悪魔ポイント (AP) を新たに定義する  
        // 悪魔ポイントはなんかこの辺の処理で使うローカル変数  
        return ap;  
    }  
}
```

このapはローカル変数（悪魔ポイント）のAPとなってしまふ.

質問に対する回答

this.について

- このようなトラブルが起こり得るので以下の運用を徹底する.
 - ★フィールドのapを指すときには「**this.**」必ずを付ける
 - ★ローカル変数のapを指すときには何もつけない

```
public class Monster{  
    int hp = 100;           // ヒットポイント  
    String name = "スライミ";  
    int ap = 10;           // 攻撃力  
  
    public int attack(){  
        int ap = 10;       // 悪魔ポイント (AP) を新たに定義する  
        // 悪魔ポイントはなんかこの辺の処理で使うローカル変数  
        return this.ap;  
    }  
}
```

こうすることで、フィールドのapを表現することが可能となる.

質問に対する回答

this.について

- 練習問題 2 は `return (int)(Math.random()*10) + this.ap` の方が良いのではないか.
 - その通り. この段階ではthisを教えていなかったためである.

クラスとインスタンス

練習問題 2 : Monsterクラスの拡張

このままだと、各モンスターは同じ程度の攻撃しかしてこなくてつまらない。Monsterクラスに、int型フィールド "ap" を定義し、attack()の戻り値はap+乱数とせよ。

```
public class Monster{  
    int hp;  
    String name;  
    int ap;  
    public int attack(){  
        //0~9までの乱数のダメージを返すメソッド  
        return (int)(Math.random()*10) + ap;  
    }  
}
```

ここを**this.ap**
とするというお話

質問に対する回答

コンストラクタについて

- 「コンストラクタで初期化を行う」の意味が分からない.

コンストラクタとは、インスタンスが生成された直後に実行される処理を記述する場所のことである.

```
public class Monster{  
    int hp = 100;  
    String name = "";  
  
    public Monster(int hp, String name){  
        this.hp = hp;  
        this.name = name;  
    }  
}
```

コンストラクタの定義

初期化処理等を記述する

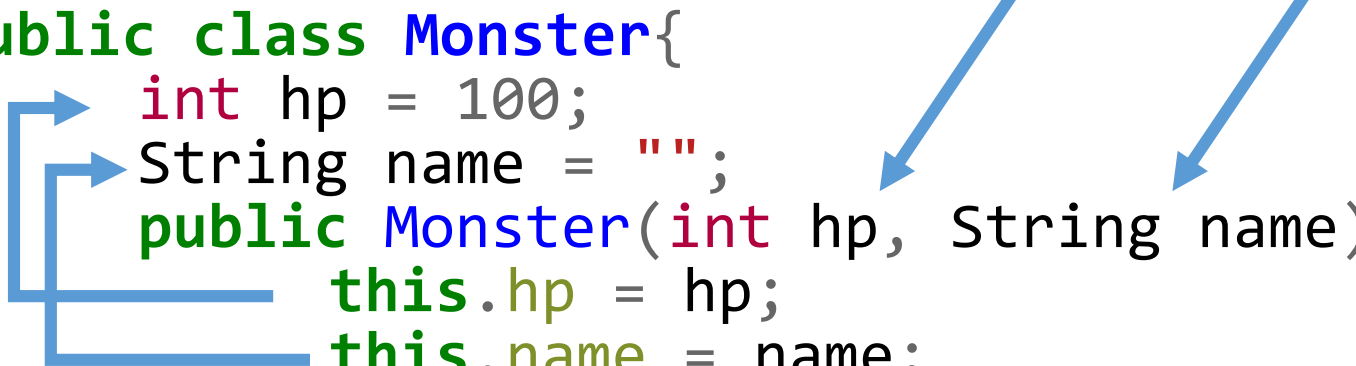
質問に対する回答

コンストラクタについて

newの時に引数を渡すと、**コンストラクタ**に引数が送られるので、ここで初期値として設定する処理を書いておくことで、毎回の代入処理を記述しなくてよくなる。

```
public static void main(String[] args){  
    Monster m1 = new Monster(100, "スライミ");  
}
```

```
public class Monster{  
    int hp = 100;  
    String name = "";  
    public Monster(int hp, String name){  
        this.hp = hp;  
        this.name = name;  
    }  
}
```



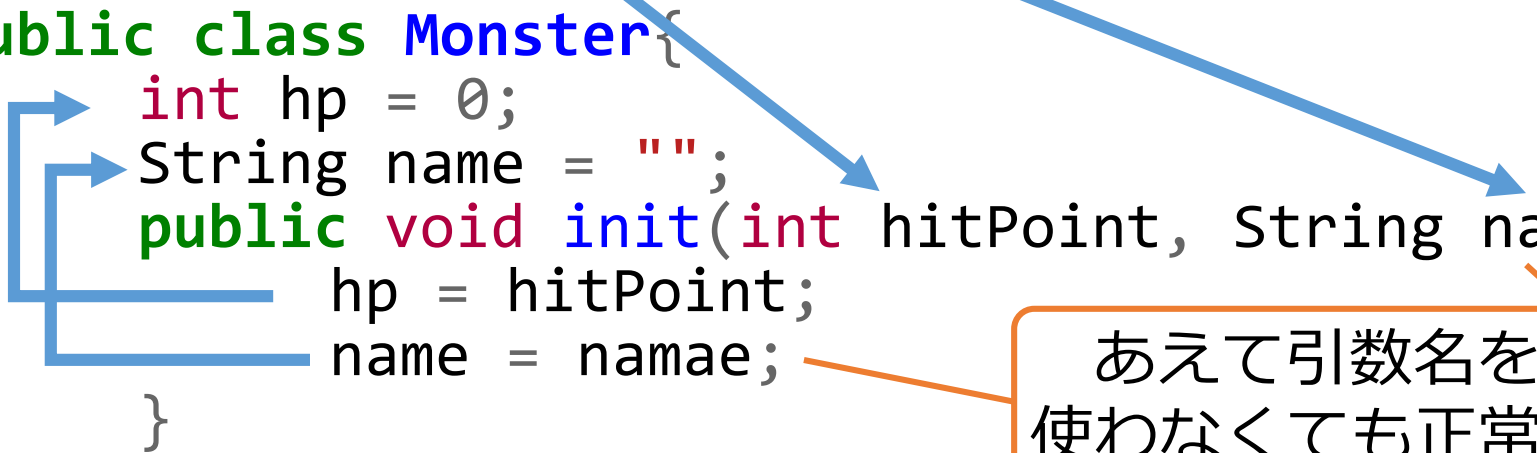
質問に対する回答

コンストラクタについて

勿論、あえて初期化用のメソッドinit()を用意してもよい。
(ただ、あえてそうする意味はない。)

```
public static void main(String[] args){  
    Monster m1 = new Monster();  
    m1.init(100, "スライムA");  
}
```

```
public class Monster{  
    int hp = 0;  
    String name = "";  
    public void init(int hitPoint, String namae){  
        hp = hitPoint;  
        name = namae;  
    }  
}
```



あえて引数名を変えるとthis.を使わなくても正常に動く（非推奨）。

質問に対する回答

コンストラクタについて

- フィールド定義時には初期化する方が良いのか.
 - コンストラクタで必ず初期化されるとも限らないので、安定動作のためには初期化する方が無難である.

```
int hp = 0;  
String name = "";
```

こういうやつ

- コンストラクタが複数定義されているとき呼び出し時にどうやって判断しているのか.
 - オーバーロードの時同様、引数の型や数でどのコンストラクタを呼ぶべきか自動で判断してくれる. 同じ引数の型や数でコンストラクタを複数定義することはできない.
- コンストラクタに可変長引数を使うことは可能か.
 - できる.

質問に対する回答

コンストラクタについて

- なぜコンストラクタを定義した後に、`new Monster()`でエラーが出たのか. (**いい質問！**)
 - `public Monster(int hp, String name)`を定義したことで、2変数のコンストラクタが定義された。それまでは定義がなかったのに`new Monster()`が実行できていた。
 - 即ち、コンストラクタが未定義の時には引数無コンストラクタが暗黙的に自動生成されており、コンストラクタを一つでも定義すると、自動定義されなくなるのである。

質問に対する回答

コンストラクタについて

- クラスと2つのコンストラクタに同じ変数を用意したとき、1つめのコンストラクタの変数を2つめのコンストラクタでどう呼び出すか。

たぶんこういう状況？

```
public class Monster{  
    int hp = 0;           // ヒットポイント (HP)  
    String name = "";  
    public Monster(){  
        // コンストラクタ 1  
        int hp = 10;     // ハッスルポイント (HP)  
    }  
    public Monster(String name){  
        // コンストラクタ 2  
        int hp = 10;     // ハンサムポイント (HP)  
    }  
}
```

質問に対する回答

コンストラクタについて

- 結論：（違うブロックの変数には）アクセスできない。

```
public class Monster{  
    int hp = 0;                //ヒットポイント (HP)  
    String name = "";  
    public Monster(){  
        int hp = 10;          //ハッスルポイント (HP)  
        this.hp               //ヒットポイントを意味する。  
        hp                   //ハッスルポイントを意味する。  
    }  
    public Monster(String name){  
        int hp = 10;          //ハンサムポイント (HP)  
        this.hp               //ヒットポイントを意味する。  
        hp                   //ハンサムポイントを意味する。  
    }  
}
```

※このソースはサンプルなのでコンパイルできない

質問に対する回答

- ランダム関数の意味が分かりづらかった.
 - `(int)(Math.random()*10)`で0~9までの乱数とした.
これは`Math.random()`が0.0以上1.0未満の乱数を生成するメソッドであるため、10掛けてintにキャストする.
- malloc的なものはあるのか.
 - mallocは動的配列確保 = プログラミングの段階で配列の数が未定の配列を確保するという話だった.
 - Javaは標準でそのような仕様になっている. 例えば,

```
int i = (int)(Math.random()*10)+1;  
int[] array = new int[i];
```

質問に対する回答

- クラスを配列にすることはできるのか？ (いい質問！)

```
//Monsterクラスのインスタンスを大量に生成
Monster[] monsters = new Monster[100];
for(int i=0;i<monsters.length;i++){
    monsters[i] = new Monster(i, "スライミ"+i);
    System.out.println(monsters[i].name);
}
```

各インスタンス
の初期値はnull

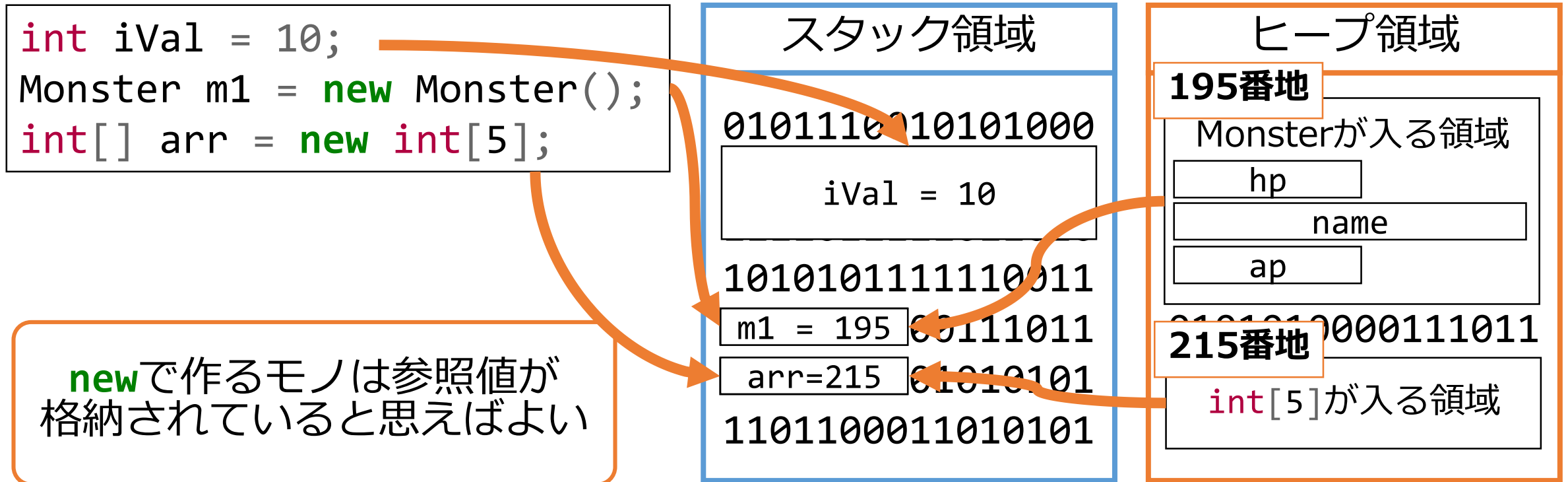


スライミ0
スライミ1
...

出力

Javaのメモリ領域

基本型（プリミティブ型）は変数領域に値が直接入る。
参照型（インスタンスや配列）は変数領域に参照値が入る。



質問に対する回答

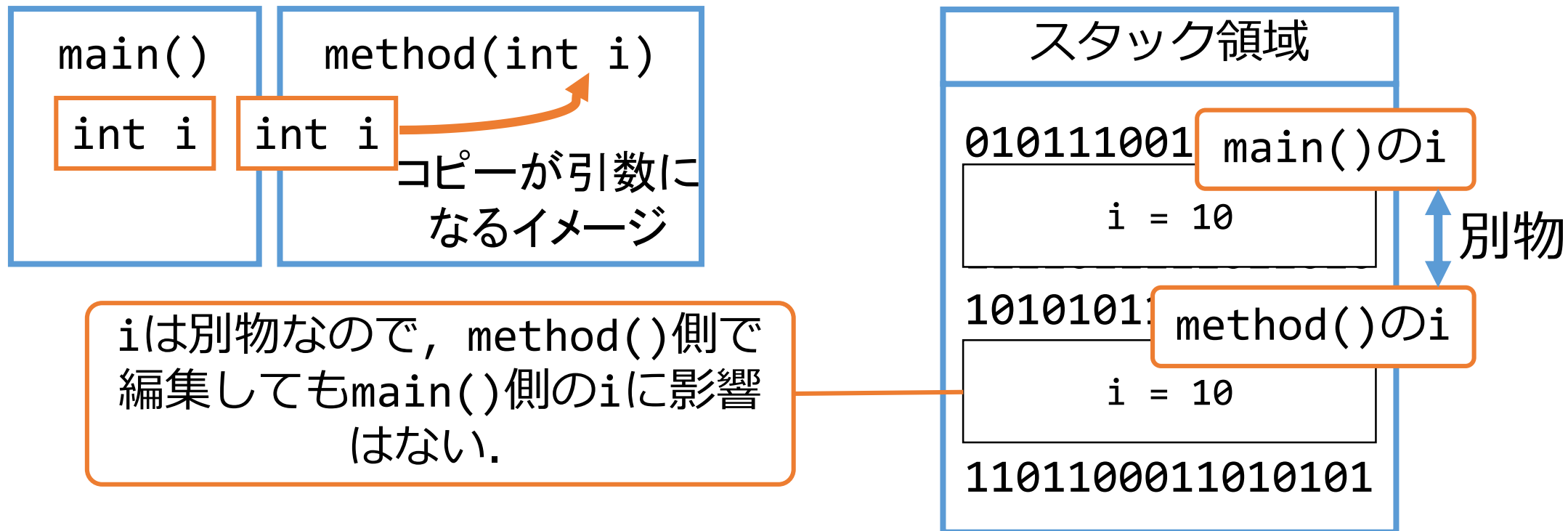
- newにはどういう意味があるのか.
 - 前の図でいうヒープメモリ側にインスタンスの領域を確保するという意味がある.
 - メモリの話が良くわからない人は, newを付けるとインスタンスや配列が作れるんだなって覚えておくといよい.
- newを付けるとC言語でいう&を付けるような動きであっているか.
 - Monster()というインスタンスの参照値を指すという意味で近いと思う.
 - ただ, 上述した通り, 本来はインスタンスを作成するという意味合いが強い.

Javaのメモリ領域

メソッドに引数を渡すときの動き

int型等，通常型の時

引数で値を送るとき，実物は送らず値だけを送るイメージ
→メソッド内で値を編集しても，呼び出し元は変わらない。

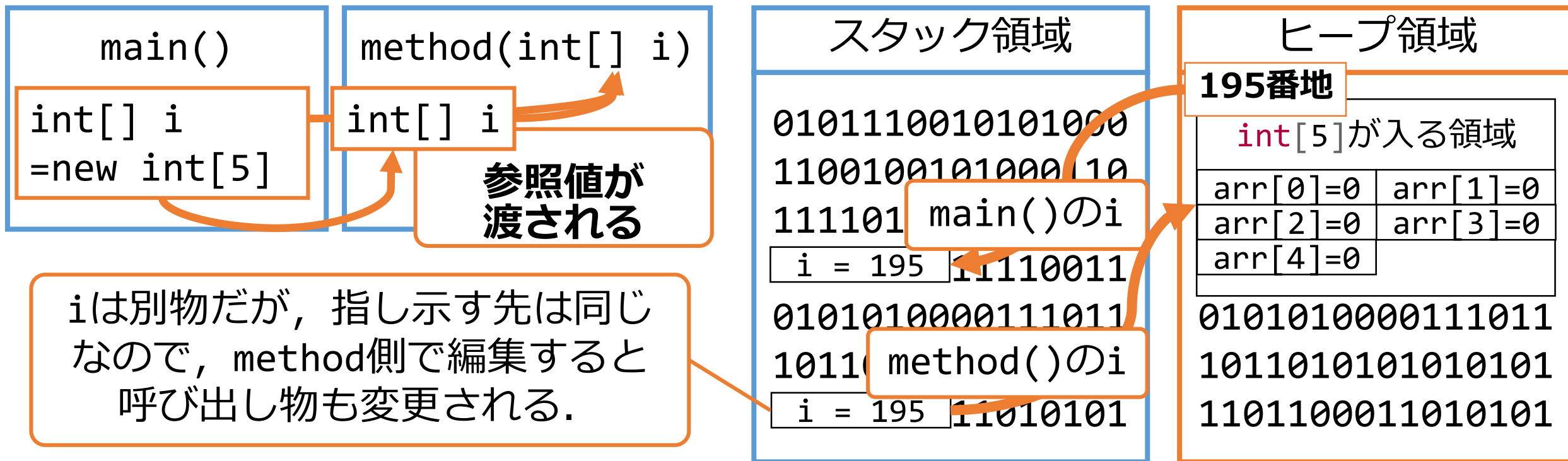


Javaのメモリ領域

メソッドに引数を渡すときの動き

配列やインスタンス等，参照型の時

引数で値を送るとき，実物をそのまま送るイメージ
→メソッド内で値を編集すると，呼び出し元も変更される.



Javaのメモリ領域

インスタンスのコピー

従って、インスタンスを代入した場合も中身はコピーされない点には注意が必要である。

```
Monster m1 = new Monster(200, "スライミ", 10);  
Monster m2 = m1;  
  
m2.name = "ドラゴヌ";  
System.out.println("m1.name = " + m1.name);
```



```
m1.name = ドラゴヌ
```

m2の名に設定したのに、
m1.nameもドラゴヌになる。

Javaのメモリ領域

インスタンスのコピー

```
public static void main(String[] args){  
    Monster m1 = new Monster(200, "スライミ", 10);  
    Monster m2 = m1;  
  
    m2.name = "ドラゴヌ";  
    System.out.println  
        ("m1.name = " + m1.name);  
}
```

m1とm2自体は別物だが、指し示す先は同じなので、m2を編集するとm1も変更される。

スタック領域

0101110010101000
1100100101000110
111101 main()のm1
m1 = 195 110011
0101010000111011
101101 main()のm2
m2=m1(195) 010101

ヒープ領域

195番地

Monsterが入る領域

hp

name

ap

0101010000111011
1011010101010101
1101100011010101

オブジェクト指向プログラミング

復習

- オブジェクト指向プログラミングとは
現実世界のモノ等のオブジェクト単位でプログラムを開発していく考え方のことである.
- 修得には訓練が必要である.
- その前に, Javaの文法や使い方の理解を頑張ろう.

質問に対する回答

オブジェクト指向について

- 企業ではオブジェクト指向をどのくらい使いこなしているのか. どこまで極めるとよいのか.
 - ピンキリである. 旧体質の企業だと, 新卒は現場見て学べという雰囲気強い印象がある. (私の偏見かも)
- オブジェクト指向の力を鍛えるにはどうすればよいか.
 - 簡単な課題をオブジェクトに分解してプログラミングする練習を繰り返し, 徐々に規模を広げていくとよい. いきなり超大規模プログラムを考えるのはたぶん無理である.
- オブジェクト指向を身に着けると他に何か役立つか.
 - 個人的には, データベースの正規化の考え方に通じるものがあると思う.

質問に対する回答

オブジェクト指向について

- C言語でオブジェクト指向が実現できるのか.
 - C言語はクラスという考え方がないので難しい.
- よく使われるクラスはネットに落ちていたりするか？
 - よく使うメソッド集などは標準APIとして準備されている.
クラスは、課題によって要求がバラバラなので、一律でこれといったものはあまりないように思う.
- C言語のポインタと今回のメモリ管理の話の違いは？
 - C言語では開発者が明示的にポインタへの処理を記述する必要があったが、Javaでは隠蔽されており、内部のメモリの動きは開発者には見えない. しかし、動きを知っておくと、Javaの動きを理解しやすいので説明している.

質問に対する回答

Java全体に対して

- Javaのクラス分けとC言語の分割コンパイルの違い.
 - Javaのクラス分けは前回説明したインスタンスとして使うことが多い. C言語は前々回説明した関数をファイル分けするという使い方である. これはJavaでも書ける.
- 聞けば聞くほどJavaが有能でCが無能.
 - 普段目に付くような部分 (GUIの画面とか) にはJava等が向いているかもしれない. 基幹部分でデータをゴリゴリ処理したい場合はCの方が高速で便利だったりする.
- Javaでゲームを作るには? アプリを作るには?
 - 今はコンソール表示だけを使っているが, Java標準にもGUIのボタンや画像を表示する機能がある. Androidだとそれ用のAPIを使って開発することでアプリ開発ができる.

質問に対する回答

Java全体に対して

- 提出課題の採点はどうなっているのか.
 - 解答はWebにUP済み. 結果何点になっているのかはメールで問い合わせてくれれば回答する.
- オブジェクト指向のおすすめのサイトや本を教えて.
 - 授業のレベルについていくために～という話であれば, 指定の参考書 (スッキリわかるJava入門 第2版) が分かりやすく書かれている.
 - Javaの文法をマスターした上でのオブジェクト指向の考え方を学ぶという意味であれば, , , , 「Java オブジェクト指向」でググると色々出てくるので, 色々読んでみるとよい. (プロIVの時に再度参考書を吟味する予定)

質問に対する回答

その他について

- 進捗が早くて大変だった。
 - 初めて習う項目は、その場ですぐに理解できないこともある。なるべく繰り返し使うように講義内では心がけるので、回を重ねるごとに理解を深めてもらえればと思う。
- スラ●ムやリュウオ●をそのまま使わないのはなぜか。
 - 念のため著作権問題を気にした。（問題ない気もするが。）
- ラインスタンプの続編は？
 - さすがにめんどい．．．（作業量の割に合わない）
- 各スタンプの使用頻度はわかるのか？
 - わかる。

本日の目次

- 復習と質問に対する回答
- **命名規則（前回の残り）**
- 修飾子（前回の残り）
- 課題演習

命名規則

変数，メソッド，クラスと一通りの説明を終えたので命名規則の一部を紹介する。

まず，Javaにおける命名の前提として以下がある。

- ・ 先頭文字に数字は使えない。
- ・ 文字数制限はない。
- ・ 大文字と小文字は区別される。
- ・ 予約語は使えない。

命名規則

クラス名

クラスの命名にはPascal記法を用いる。

Pascal記法は、以下3点の規則で成り立つ。

- ・ 先頭を大文字にする。
- ・ それ以外は小文字にする。
- ・ 単語の区切りは大文字にする。

例えば、

- ・ ArrayList
- ・ HashMap
- ・ SimpleDateFormat

命名規則

メソッド名, 変数名

メソッドと変数の命名にはcamelCase記法を用いる.
camelCase記法は, 以下3点の規則で成り立つ.

- 先頭を小文字にする.
- それ以外は小文字にする.
- 単語の区切りは大文字にする.

例えば,

- getTime()
- userName
- currentTimeMillis()

本日の目次

- 復習と質問に対する回答
- 命名規則（前回の残り）
- **修飾子（前回の残り）**
- 課題演習

修飾子

final

finalは、定数を宣言するための修飾子である。

```
final [変数型] [変数名] = [定数];  
> final int MAX_HP = 20000;
```

finalを付けると、変数のように読み出しはできるが、値の変更はできなくなる。

final定数の命名規則はアンダーバー区切りの全て大文字を使用する。

Cでいう#defineに近い。

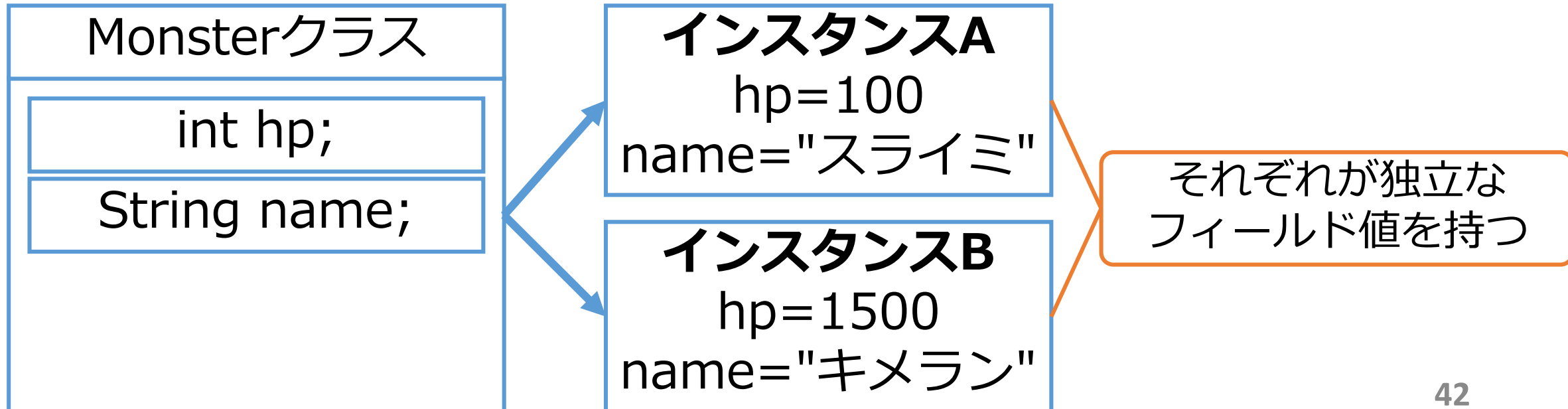


修飾子

static

staticは、静的メンバを宣言するための修飾子である。

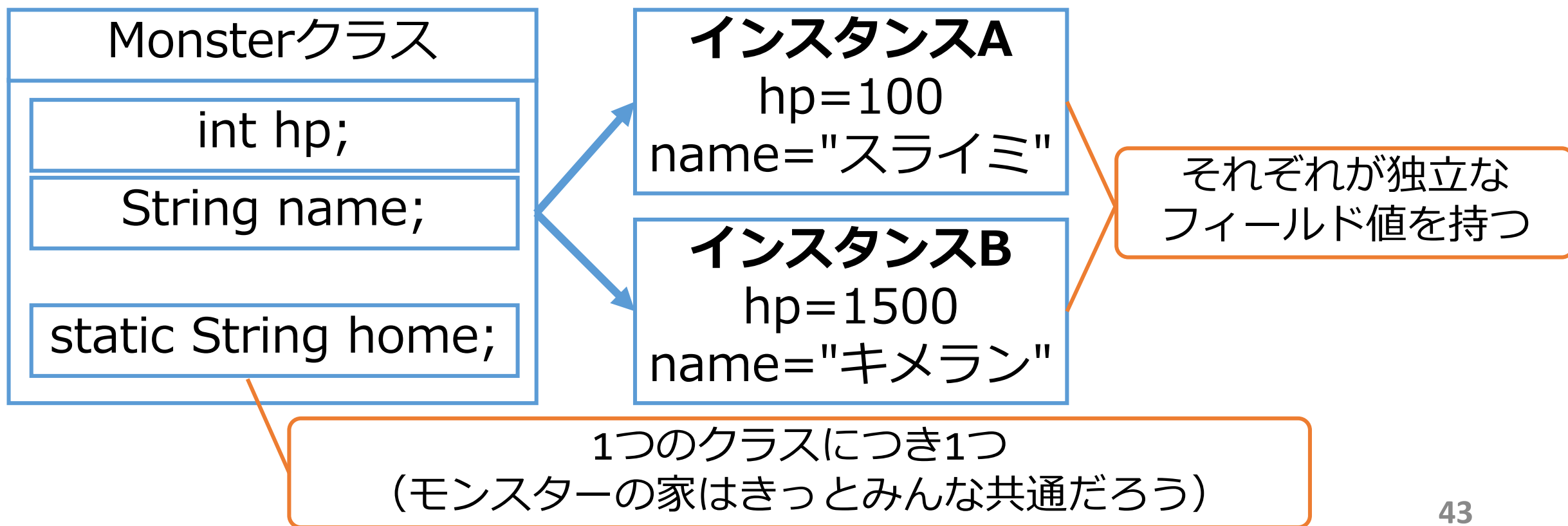
今回の講義で取り扱ってきたフィールドは全て**static**がついていない。**static**がついていない場合それぞれのインスタンスで独立なフィールド値を持つ。



修飾子

static

一方**static**がついているフィールドは静的メンバと呼ばれ、1つのクラスにつき1つ固有のフィールドしか持たない。



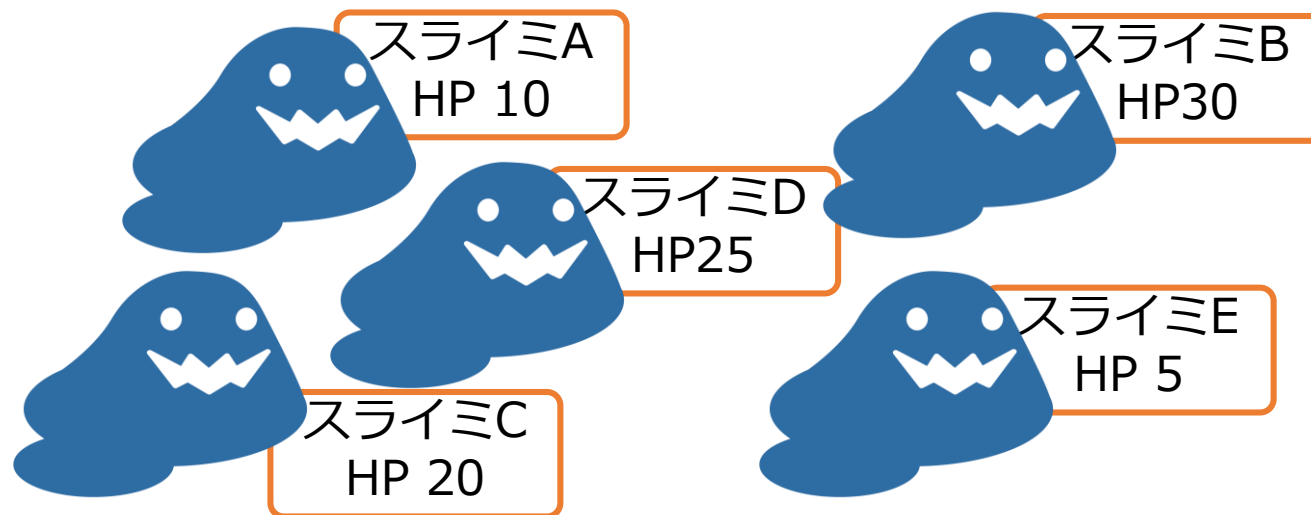
修飾子

static

クラス（金型）



インスタンス（量産されるイメージ）



staticフィールドは
クラスが唯一保持

```
public class Monster{  
    int hp = 100;  
    String name = "";  
    static String home = "";  
}
```

一般フィールドは個々
のインスタンスが保持

修飾子

static

アクセスするには、**[クラス名].[static変数名]**とするか、**[インスタンス].[static変数名]**とする。

```
Monster m1 = new Monster(200, "スライミ", 10);  
Monster m1 = new Monster(1500, "キメラン", 50);  
Monster.home = "魔王の城";  
System.out.println(Monster.home);  
m1.home = "魔物の巣窟";  
System.out.println(m2.home);
```

[クラス名].[変数名]で
アクセスすることが**標準**

m1.homeを編集しても、homeは
唯一なのでm2.homeも変わる。
勿論、Monster.homeも変わる。

魔王の城
魔物の巣窟

次週予告

※次週以降も計算機室

もう少し発展的なクラスの使い方
について学習する.



本日の目次

- 復習と質問に対する回答
- 命名規則（前回の残り）
- 修飾子（前回の残り）
- **課題演習**

本日の提出課題

課題

演習課題をEclipseで開発し,
ソースコードを提出する.

資料を公開しているサイトから, 「課題提出」で提出ページに行ける.

<http://hsgw-nas.fuis.u-fukui.ac.jp/lecture.html>

直リンクはこちら

http://hsgw-nas.fuis.u-fukui.ac.jp/lecture_file.html

質問に対する回答

- Monsterクラスを確りと作ってみたい.
- Weaponを装備したのでapをattack()に反映させたい.
- Weapon型の説明がいまいちよくわからなかった.
- クラスの中にインスタンスを持つのはわかりにくい.
- setSoubiとgetSoubiがよくわからない.

→Monsterクラスを発展させる課題にする.
(本当は別の問題を作っていたが. . .)



演習課題

次の3ファイルを提出する.

- Monster.java (演習課題 1 ~ 9 のできたところまで)
- Weapon.java (演習課題 4)
- Main.java (演習課題 7)
 - Main.java内には演習課題 7 のmain()以外は実装しない.
 - デバッグ時には色々書き換えてよいが, 提出時には演習課題 7 の要求に合わせておくようにする.

提出時にはエラーは解決しておくこと. エラーのまま提出すると全課題の採点ができない.

感想や要望, 苦情等はMain.javaのkanso()でprintln()する.

演習課題 1

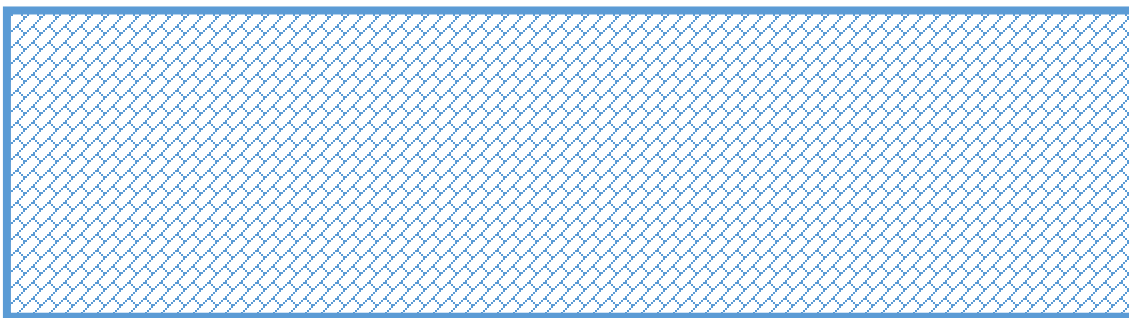
Monsterクラス

※動作確認例で正しくても、他のパターンでNGな場合×なことがある。

動作確認例：

```
public class Main {  
    public static void main(String[] args){  
        Monster m = new Monster(10,"スラ",1);  
        System.out.println(m.name);  
    }  
}
```

```
public class Monster {
```



```
}
```

課題要件：

- 敵キャラを管理するMonsterクラスを開発せよ。（前回資料と同じ部分が多いが）
- フィールドは次の3つとする。
 - int hp （初期値は0）
 - String name （初期値は空）
 - int ap （初期値は0）
- hp,name,apをまとめて初期化するためのコンストラクタを定義する。
- hp,nameだけでも初期化できる用のコンストラクタも定義する。

入出力例：

出力：スラ

演習課題 2

Monsterクラス

動作確認例：

```
public class Main {  
    public static void main(String[] args){  
        Monster m1=new Monster(10,"スラ",1);  
        Monster m2=new Monster(50,"キメ",5);  
        System.out.println(m1.toString());  
        System.out.println(m2.toString());  
    }  
}
```

```
public class Monster {  
      
}
```

課題要件：

- デバッグ用にフィールドを全て結合しStringで返すメソッドtoString()を開発せよ.
- 出力順はhp,name,apとする.
(入出力例を参照)

入出力例：

出力：10,スラ,1
50,キメ,5

演習課題 3

Monsterクラス

動作確認例：

```
public class Main {  
    public static void main(String[] args){  
        Monster m1=new Monster(10,"スラ",1);  
        System.out.println(m1.attack()  
            +"のダメージを受けた");  
    }  
}
```

```
public class Monster {  
      
}
```

課題要件：

- Monsterからのダメージを取得するメソッドattack()を開発せよ.
- ダメージは各Monsterインスタンスのapと同値とする.

入出力例：

出力：1のダメージを受けた

演習課題 4

Monsterクラス + Weaponクラス

動作確認例：

```
public class Main {  
    public static void main(String[] args){  
        Monster m = new Monster(10, "スラ", 1);  
        Weapon w = new Weapon("ただの板", 3);  
        m.setSoubi(w);  
        System.out.println(m.soubi.name);  
    }  
}
```

```
public class Monster {  
      
}
```

```
public class Weapon {  
      
}
```

課題要件：

- 武器を管理するWeaponクラスを開発せよ.
- フィールドは次の2つとする.
 - String name (初期値は空)
 - int ap (初期値は0)
- name, apをまとめて初期化するためのコンストラクタを定義する.
- MonsterクラスのフィールドにWeapon型のインスタンスsoubiを追加せよ.
- soubiを変更するためのメソッドsetSoubi(Weapon soubi)を実装せよ.

入出力例：

出力：ただの板

演習課題 5

Monsterクラス + Weaponクラス

動作確認例：

```
public class Main {  
    public static void main(String[] args){  
        Monster m = new Monster(10, "スラ", 1);  
        Weapon w = new Weapon("ただの板", 3);  
        m.setSoubi(w);  
        System.out.println(m.attack2()  
                             + "のダメージを受けた");  
    }  
}
```

```
public class Monster {  
      
}
```

課題要件：

- 装備した武器に応じてMonsterインスタンスからの攻撃力が変わるようにattack()を改造したい。ただ、採点の都合上、attack2()としてこれを実装せよ。
- attack2()のダメージは各Monsterインスタンスのapに、装備している武器のapを加算したものとする。
- ただしsoubiがnullの場合、通常攻撃と同じダメージとすること。

入出力例：

出力：4のダメージを受けた

演習課題 6

Monsterクラス + Weaponクラス

動作確認例：

```
public class Main {  
    public static void main(String[] args){  
        Monster m = new Monster(10, "スラ", 1);  
        m.damage(1);  
        m.damage(10);  
    }  
}
```

```
public class Monster {  
      
}
```

課題要件：

- Monsterインスタンスにダメージを与えて倒せるよう、damage(int val)を開発せよ.
- ダメージを与えるとvalの値だけhpが減少する.
- メソッドを実行すると減少後のhpを入出力例2行目のように出力する.
- hpが0以下になった場合hpは0とし、入出力例3行目のように出力を変える.
- スラや1のダメージ、残りHP等はフィールド値によって変化する.

入出力例：

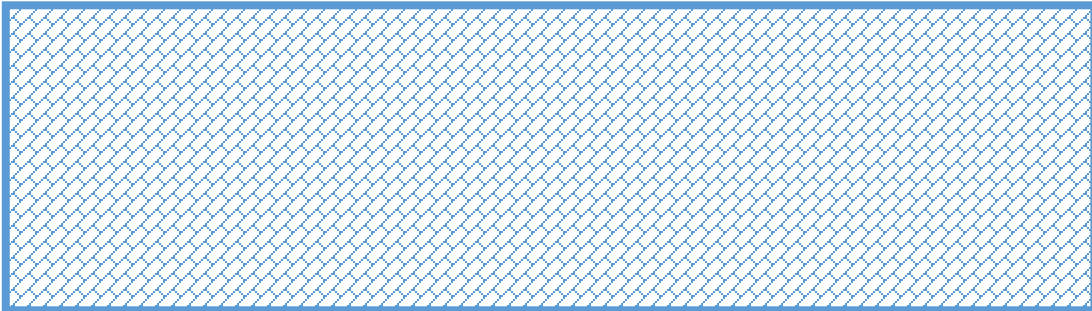
出力：スラに1のダメージ！残りHP9

スラに1のダメージ！スラを倒した！

演習課題 7

Monsterクラス + Weaponクラス

動作確認例：

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

課題要件：

- main()側の操作の練習をしてほしい.
- 2体のMonsterインスタンス, 1つのWeaponインスタンスを作成する.
Mons:hp 50 name キメラン ap 10
Mons:hp 100 name ドラゴヌ ap 100
Weap:name ドラゴヌの牙 ap 50
- ドラゴヌの牙をドラゴヌに装備する.
- その後, 入出力例の出力となるように適宜main()を記述してほしい.

出力：

キメランとドラゴヌが現れた！
キメラン残りHP50！ドラゴヌ残りHP100！
ドラゴヌからの攻撃！150のダメージを受けた！
キメランに100のダメージ！キメランを倒した！

(printlnを使用)
(printlnを使用)
(printlnを使用)
(メソッドを使用)

演習課題 8

Monsterクラス + Weaponクラス

動作確認例：

```
public class Main {  
    public static void main(String[] args){  
        Monster m = new Monster(10, "スラ", 1);  
        Weapon w = new Weapon("ただの板", 3);  
        m.setSoubi(w);  
        System.out.println(m.attack3()  
                             + "のダメージを受けた");  
  
        m.damage(7);  
        System.out.println(m.attack3()  
                             + "のダメージを受けた");  
    }  
}
```

```
public class Monster {  
      
}
```

課題要件：

- Monsterはhpが5以下になると、火事場の馬鹿力で攻撃力が2倍 + 装備した武器の攻撃力になるようにattack()を改造したい。ただ、採点の都合上、attack3()としてこれを実装せよ。

入出力例：

出力：4のダメージを受けた
5のダメージを受けた

演習課題 9

Monsterクラス + Weaponクラス

動作確認例：

```
public class Main {  
    public static void main(String[] args){  
        Monster m = new Monster(10, "スラ", 1);  
        m.setSoubi(new Weapon("ただの板", 3));  
        System.out.println(m.attack4()  
                             + "のダメージを受けた");  
        m.damage(7);  
        System.out.println(m.attack4()  
                             + "のダメージを受けた");  
        System.out.println(m.attack4()  
                             + "のダメージを受けた");  
    }  
}
```

```
public class Monster {  
      
}
```

課題要件：

- Monsterはhpが10未満になると、50%の確率でhpを10回復するようにattack()を改造したい。ただ、採点の都合上、attack4()としてこれを実装せよ。
- なお、回復時にはattack4()の戻り値は0とし、その代わりに「スラはHPを10回復し13となった。」のような出力を行うこととする。

出力例：4のダメージを受けた

スラに7のダメージ！残りHP3

スラはHPを10回復し13となった

0のダメージを受けた

4のダメージを受けた

その他

hb160692の人の前回コメントに

- 常にeclipseで開く設定にしてしまい、どうかえればよいのかがわからない。

とあったが解決済みか？質問して.