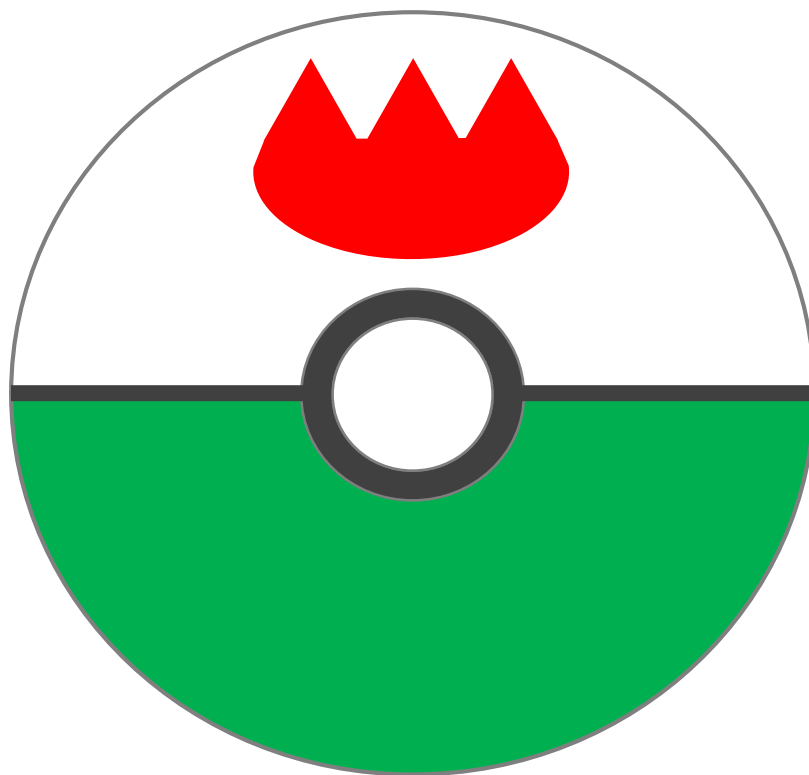




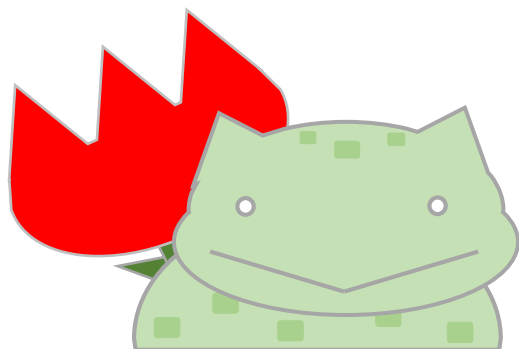
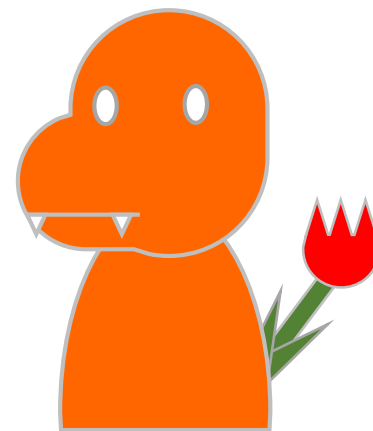
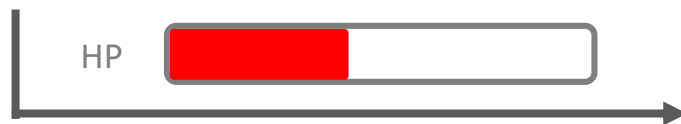
ウズウズカレツジ プログラマーコース

総合演習（モンスター対戦ゲームの作成）

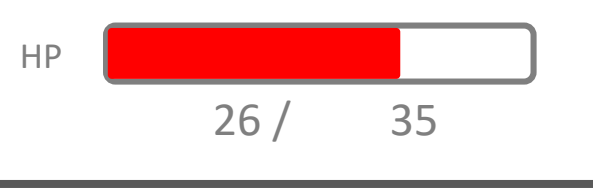
チューリップモンスター (仮)



ヒトカケ Lv 12



フシギヤネ Lv 9



▶ たたかう にげる



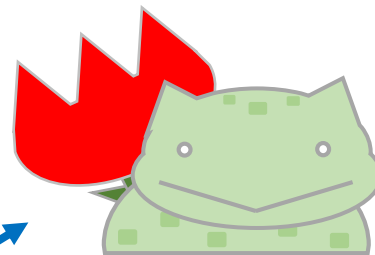
SimulateMonsterBattle
クラス

インスタンス化



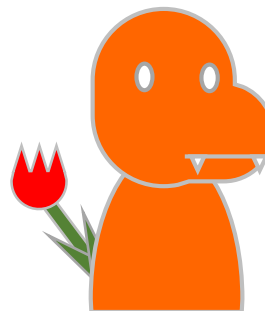
BattleWild
クラス

インスタンス化



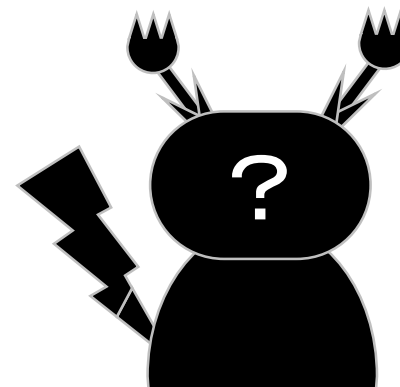
Fushigiyade
クラス

インスタンス化



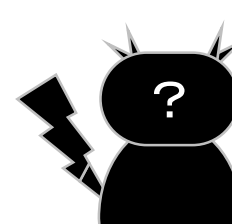
Hitokake
クラス

継承

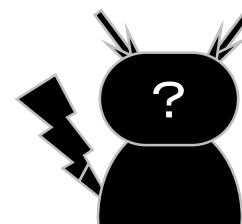


Monster3
クラス

継承



Monster1
クラス



Monster2
クラス

churimon
パッケージ

提供ソースコード



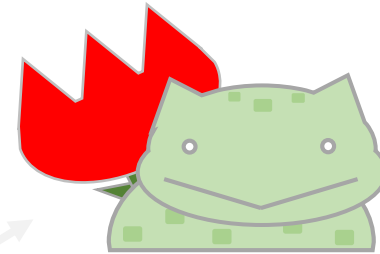
churimon
パッケージ

インスタンス化

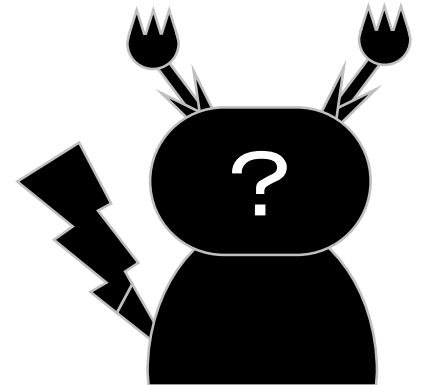
インスタンス化

継承

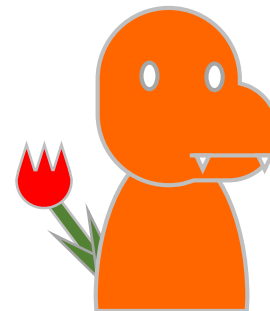
継承



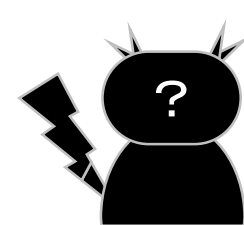
Fushigiyade
クラス



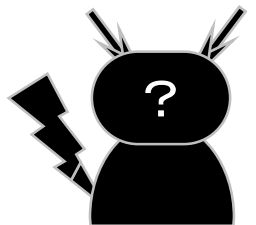
Monster3
クラス



Hitokake
クラス



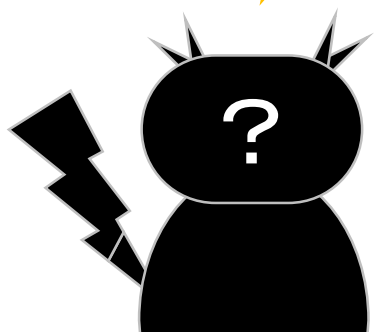
Monster1
クラス



Monster2
クラス

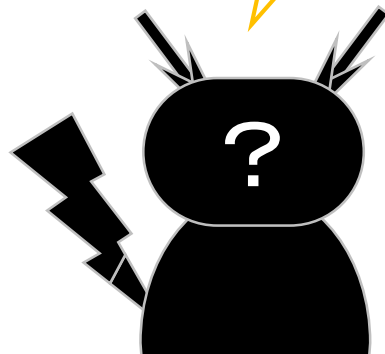
作ってみましょう！

基本的なメソッド



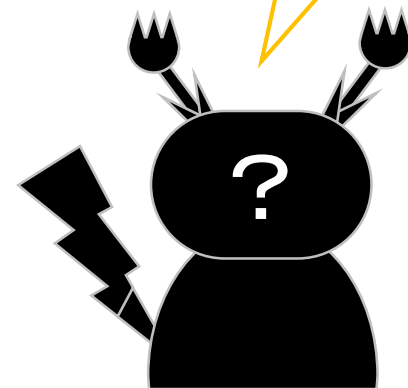
Monster1クラス

基本的なメソッド
コンストラクタ



Monster2クラス

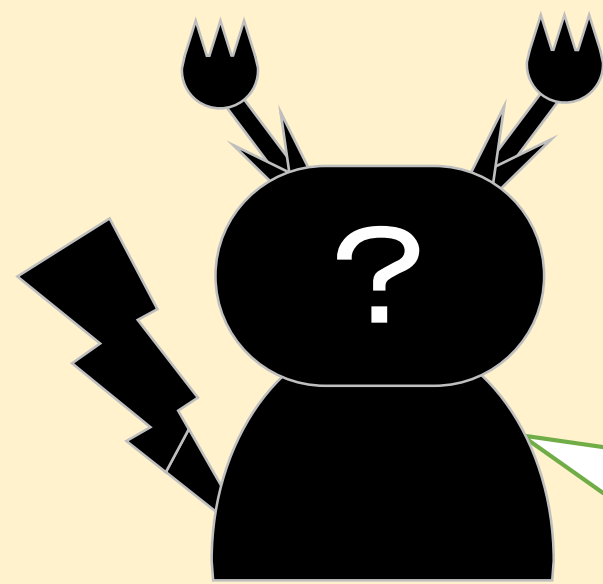
基本的なメソッド
コンストラクタ
カプセル化



Monster3クラス

< 演習（ゲーム作成 その1） >
「Monster1.java」を作成します。

(1) 下記に従ってフィールドを作成してください。



◆フィールド（モンスターの情報）

型	フィールド名	初期値	補足
String	character	(unknown)	種族
String	trainer	(wild)	トレーナー
String	name	(noname)	なまえ
int	lv	1	レベル
int	hp	80	HP
int	atk	15	こうげき
int	def	10	ぼうぎょ
int	spd	10	すばやさ
int	hpMax	80	HP初期値
String	wazaNm	たいあたり	わざ(なまえ)
String	wazaDmgRate	1.0	わざ(ダメージ倍率)

<演習（ゲーム作成 その1）>

「Monster1.java」を作成します。

(2) 下記に従ってjava.lang.ObjectクラスのtoStringメソッドをオーバーライドしてください。

→この先メソッドの動作確認などで使用していきましょう



[toStringメソッド]

toStringメソッドのオーバーライドを以下のように実施します。

※修飾子publicをつけること

・以下のフィールドの値を結合した文字列を返します。

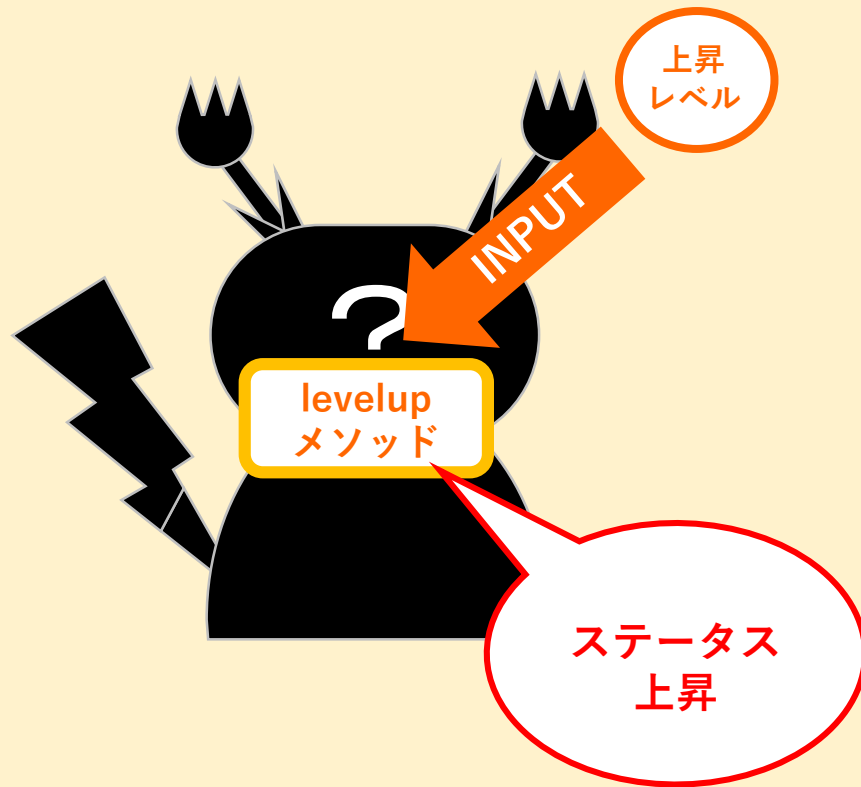
- character
- trainer
- name
- lv
- hp
- atk
- def
- spd
- hpMax
- wazaNm
- wazaDmgRate

【例】

<フィールド確認> character:ヒトカケ / trainer:ぼく / name:カケ郎 / lv:21 / hp:660 / atk:175 / def:110 / spd:190 / hpMax:660 / wazaNm:たいあたり / wazaDmgRate:1.0

< 演習（ゲーム作成 その1） >
「Monster1.java」を作成します。

(3) 下記に従ってlevelUpメソッドを作成してください。



[levelUpメソッド]

上昇レベルに従ってステータスを上昇させます。

<引数>

上昇レベル (int型)

<戻り値>

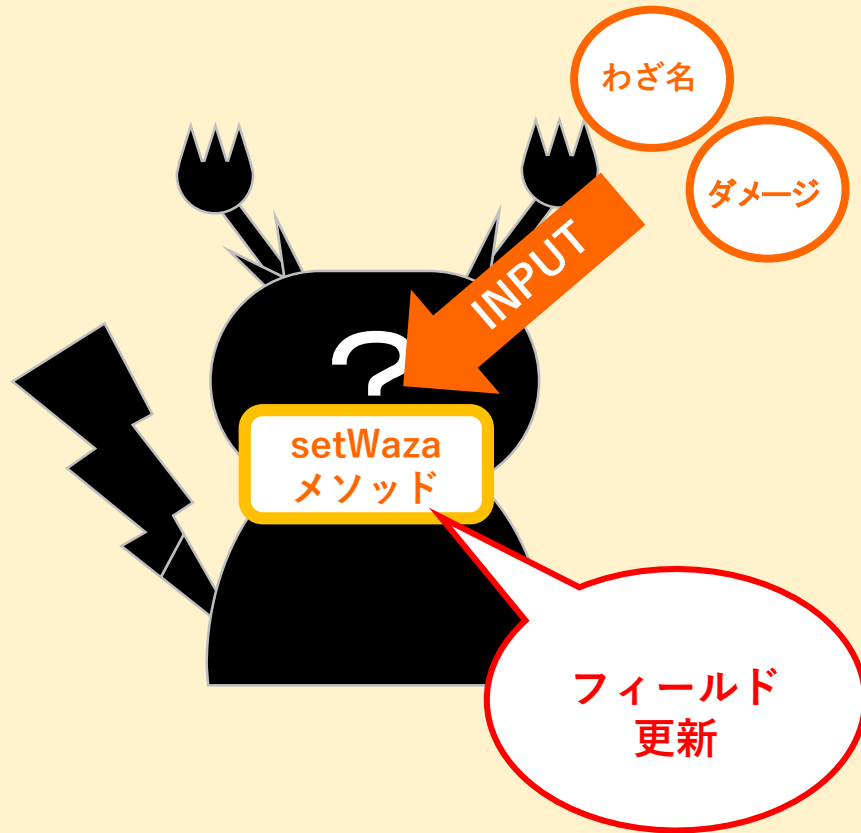
なし

<機能詳細>

- ・ 引数を元に一部フィールドの値を更新します。（以下詳細）
 - lv : 上昇レベル×1の値を既存値にプラスする
 - hpMax : 上昇レベル×30の値を既存値にプラスする
 - atk : 上昇レベル×5の値を既存値にプラスする
 - def : 上昇レベル×5の値を既存値にプラスする
 - spd : 上昇レベル×5の値を既存値にプラスする
 - hp : 更新後のhp_maxの値を代入する

< 演習（ゲーム作成 その1） >
「Monster1.java」を作成します。

(4) 下記に従ってsetWazaメソッドを作成してください。



[setWazaメソッド]
わざに関する情報を設定します。

<引数>

引数1:わざ名 (String型) ,引数2:わざのダメージ倍率 (String型)

<戻り値>

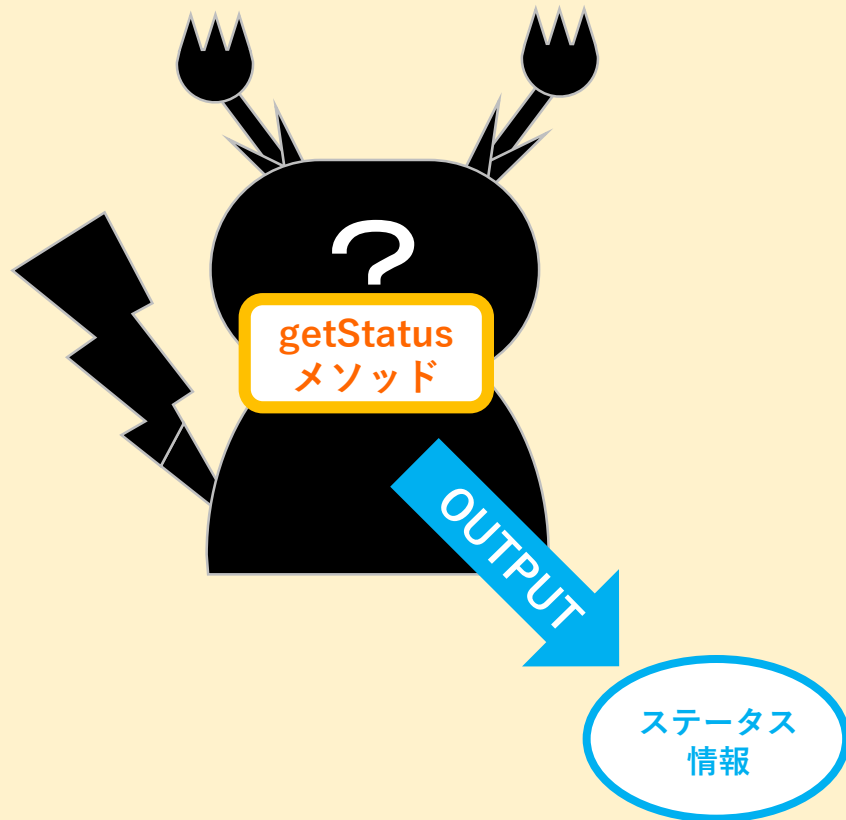
なし

<機能詳細>

- ① 引数2のバリデーションチェックを行います。
チェック内容：引数2が「X…X.X(Xは0～9の数字のいずれか)」形式か
※ matchesメソッド (Stringクラス) を使用し、正規表現を用いた
チェックを行う
※チェックで用いる正規表現： `^[0-9]+¥.[0-9]$`
- ② ①のチェックがOKであれば引数を元に一部フィールドの値を更新します。(以下詳細)
 - wazaNm： わざ名 (引数1) の値を代入する
 - wazaDmgRate： ダメージ (引数2) の値を代入する
- ③ ①のチェックがNGであればフィールドの更新は行わずに
エラーメッセージ「[ERROR]わざの設定に失敗しました」を画面出力
します。

<演習（ゲーム作成 その1）>
「Monster1.java」を作成します。

(5) 下記に従ってgetStatusメソッドを作成してください。



[getStatusメソッド]
ステータスを表示します。

<引数>

なし

<戻り値>

ステータス情報（String型）

<機能詳細>

①ステータス情報（一部フィールドの情報）を文字列で返します。
[(nameの値) lv(lvの値) HP(hpの値)/(hpMaxの値)]

【例】 [ピカ丸 lv20 HP500/688]

< 演習（ゲーム作成 その1） >
「Monster1.java」を作成します。

(6) 下記に従ってuseWazaメソッドを作成してください。



[useWazaメソッド]

わざを使用して相手にダメージを与えます。

<引数>

なし

<戻り値>

相手に値渡しするダメージ（int型）

<機能詳細>

①相手に値渡しするダメージを下記ルールで求めます。

相手に値渡しするダメージ：こうげき×わざのダメージ倍率

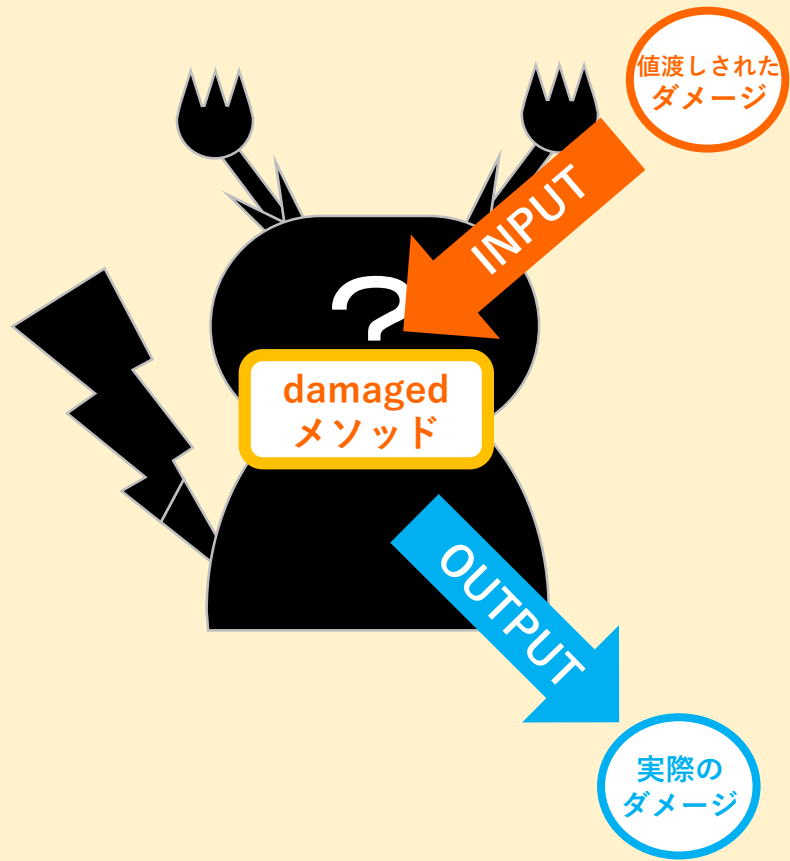
※BigDecimalを使用します。

※BigDecimal型からint型へ変換（同時に小数点以下を切り捨て）する際はintValueメソッドを使用しましょう。

② ①で求めたダメージを返します。

<演習（ゲーム作成 その1）>
「Monster1.java」を作成します。

(7) 下記に従ってdamagedメソッドを作成してください。



[damagedメソッド] その1

値渡しされたダメージから実際に受けるダメージを計算し、HPから減算します。
戻り値として実際に受けるダメージを返します。

<引数>

値渡しされたダメージ (int型)

<戻り値>

実際に受けるダメージ (int型)

<機能詳細>

①ダメージ減算率を下記ルールで求めます。

ダメージ減算率： $1 / (1 + \text{ぼうぎょ} \div 120)$ ※小数第3位切り捨て

※BigDecimalを使用します。

②実際に受けるダメージを下記ルールで求めます。

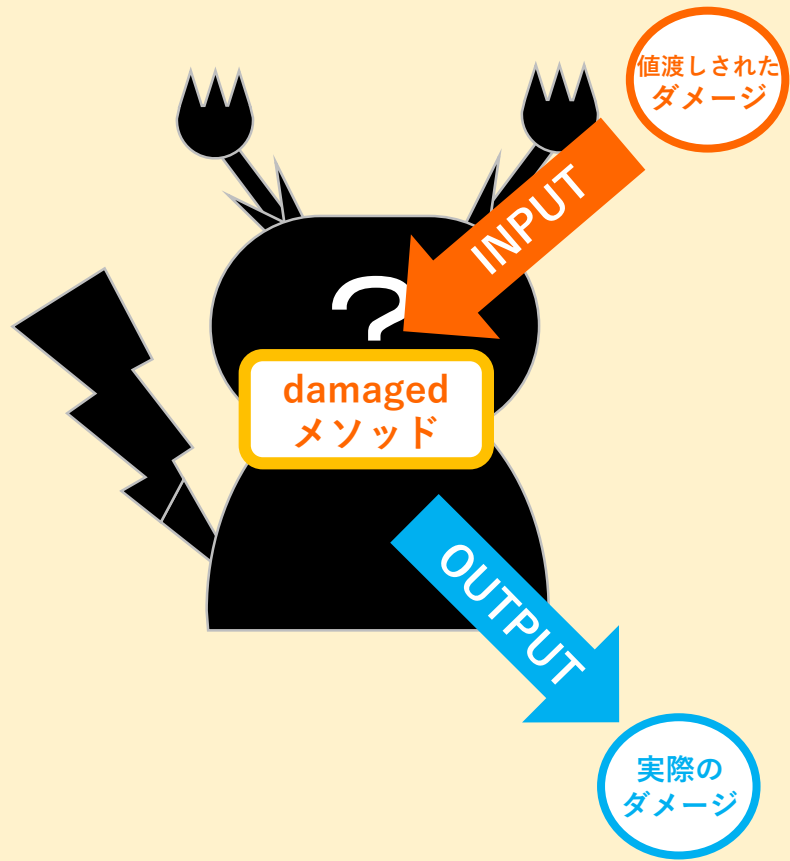
実際に受けるダメージ：値渡しされたダメージ値 × ダメージ減算率

※BigDecimalを使用します。

※useWazaメソッドをint型への変換はintValueメソッドを使用しましょう。

< 演習（ゲーム作成 その1） >
「Monster1.java」を作成します。

(7) 下記に従ってdamagedメソッドを作成してください。



[damagedメソッド] その2

値渡しされたダメージから実際に受けるダメージを計算し、HPから減算します。
戻り値として実際に受けるダメージを返します。

- ③ HPと受けるダメージを比べ、 $HP > \text{ダメージ}$ であればダメージを差し引いた値をHPに代入します。 $HP < \text{ダメージ}$ であればHPに0を代入します。
- ④ 戻り値として実際に受けるダメージの値を返します。

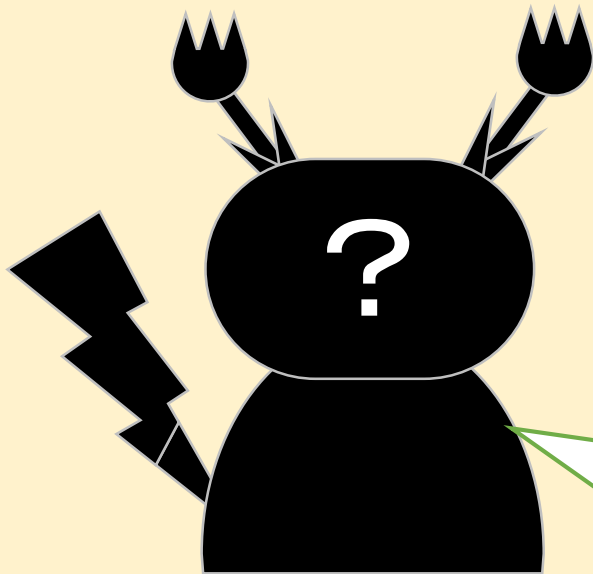
<演習（ゲーム作成 その2）>

「Monster2.java」を作成します。

ただし、メソッドはMonster1.javaと全く同じものを記述することとします。

（Monster1.javaをコピーして作成しましょう）

(1) 下記に従ってフィールドを作成してください。



◆フィールド（モンスターの情報）

型	フィールド名	初期値	補足
String	character	設定なし	種族
String	trainer	設定なし	トレーナー
String	name	設定なし	なまえ
int	lv	設定なし	レベル
int	hp	設定なし	HP
int	atk	設定なし	こうげき
int	def	設定なし	ぼうぎょ
int	spd	設定なし	すばやさ
int	hpMax	設定なし	HP初期値
String	wazaNm	設定なし	わざ(なまえ)
String	wazaDmgRate	設定なし	わざ(ダメージ倍率)

< 演習（ゲーム作成 その2） >
「Monster2.java」を作成します。

(2) 下記に従ってコンストラクタ1を作成してください。



[コンストラクタ1（引数なし）]
フィールドの初期化を行います。

<引数>

なし

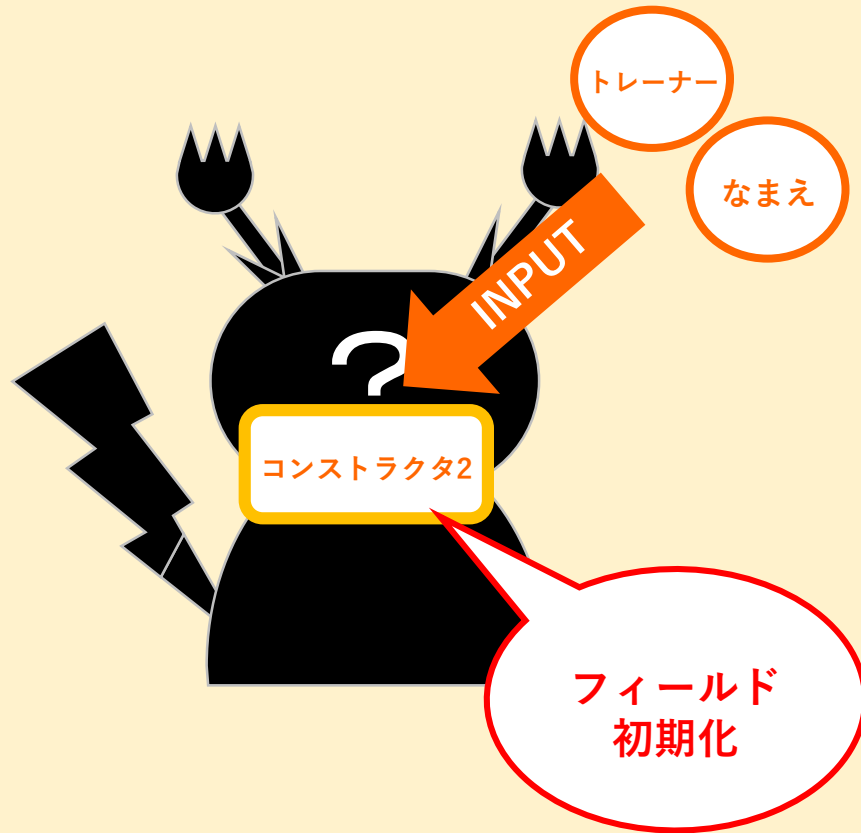
<機能詳細>

・ 以下のようにフィールドを初期化します。

- character	: (unknown)
- trainer	: (wild)
- name	: (noname)
- lv	: 1
- hp	: 80
- atk	: 15
- def	: 10
- spd	: 10
- hpMax	: 80
- wazaNm	: たいあたり
- wazaDmgRate	: 1.0

< 演習（ゲーム作成 その2） >
「Monster2.java」を作成します。

(3) 下記に従ってコンストラクタ2を作成してください。



[コンストラクタ2（引数2つ）]
フィールドの初期化を行います。

<引数>

引数1: トレーナー（String型）, 引数2: なまえ（String型）

<機能詳細>

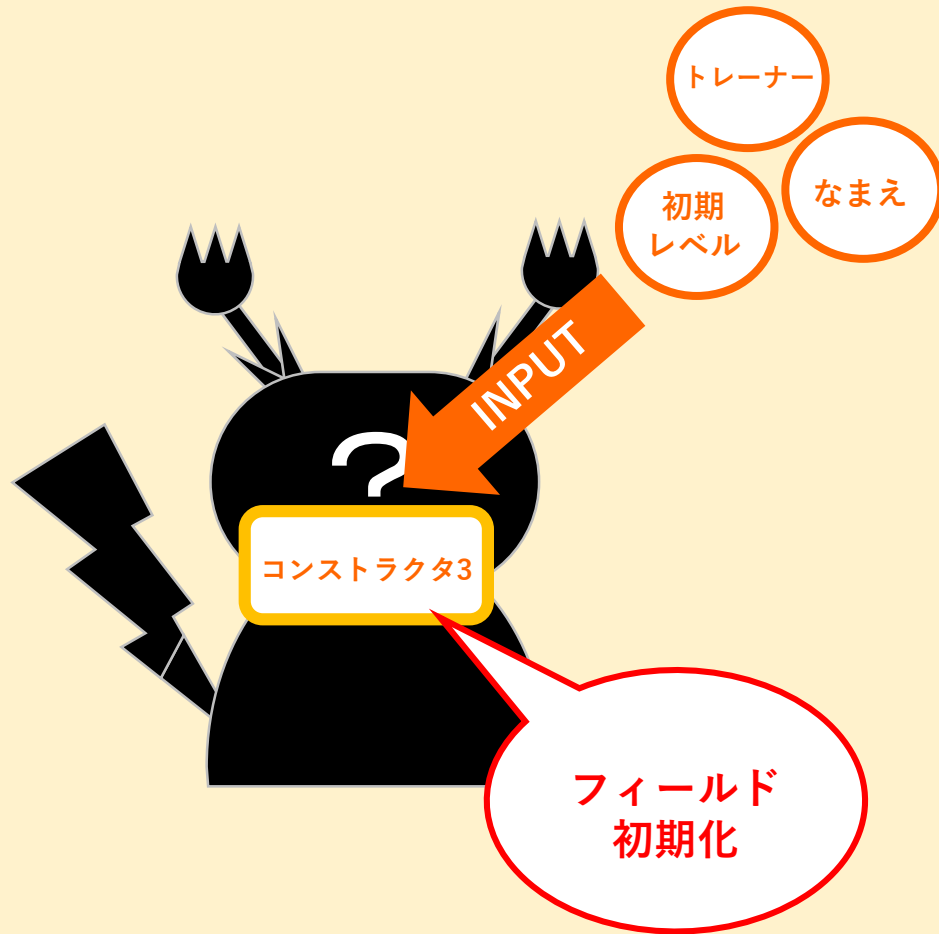
・ 以下のようにフィールドを初期化します。

- character	: (unknown)
- trainer	: 引数1（トレーナー）
- name	: 引数2（なまえ）
- lv	: 1
- hp	: 80
- atk	: 15
- def	: 10
- spd	: 10
- hpMax	: 80
- wazaNm	: たいあたり
- wazaDmgRate	: 1.0

・ コンストラクタ1を利用しましょう。

< 演習（ゲーム作成 その2） >
「Monster2.java」を作成します。

(4) 下記に従ってコンストラクタ3を作成してください。



[コンストラクタ3（引数3つ）]
フィールドの初期化を行います。

<引数>

引数1: トレーナー（String型）, 引数2: なまえ（String型）,
引数3: 初期レベル（int型）

<機能詳細>

・ 以下のようにフィールドを初期化します。

- character	: (unknown)
- trainer	: 引数1（トレーナー）
- name	: 引数2（なまえ）
- lv	: 引数3（初期レベル）
- hp	: レベルに見合った値
- atk	: レベルに見合った値
- def	: レベルに見合った値
- spd	: レベルに見合った値
- hpMax	: hpと同じ値
- wazaNm	: たいあたり
- wazaDmgRate	: 1.0

・ コンストラクタ2を利用しましょう。

・ levelUpメソッドを利用しましょう。

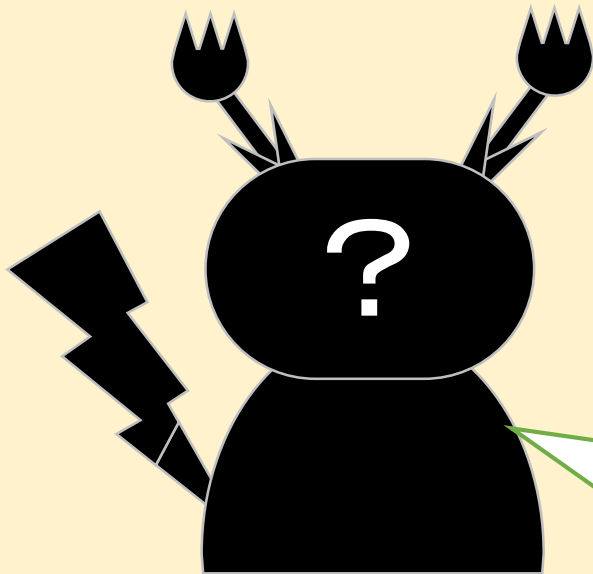
※レベルが1の場合と2以上の場合とで場合分けしましょう！

< 演習（ゲーム作成 その3） >

「Monster3.java」を作成します。

（Monster2.javaをコピーして作成しましょう）

(1) 下記に従ってフィールドを作成してください。



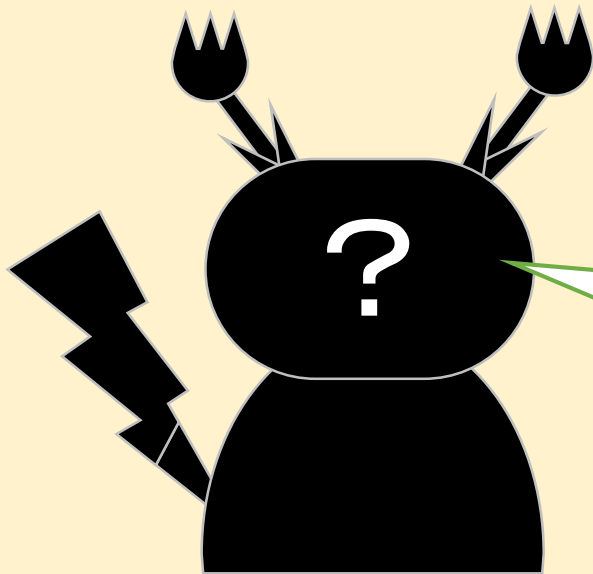
◆フィールド（モンスターの情報）

修飾子	型	フィールド名	初期値	補足
private	String	character	設定なし	種族
private	String	trainer	設定なし	トレーナー
private	String	name	設定なし	なまえ
private	int	lv	設定なし	レベル
private	int	hp	設定なし	HP
private	int	atk	設定なし	こうげき
private	int	def	設定なし	ぼうぎょ
private	int	spd	設定なし	すばやさ
private	int	hpMax	設定なし	HP初期値
private	String	wazaNm	設定なし	わざ(なまえ)
private	String	wazaDmgRate	設定なし	わざ(ダメージ倍率)

< 演習（ゲーム作成 その3） >

「Monster3.java」を作成します。

(2) 下記に従ってメソッドに修飾子を付与してください。

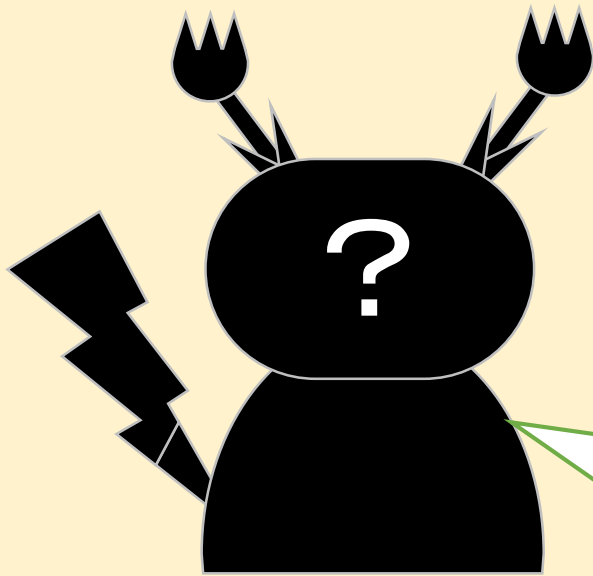


◆メソッド

- ・ 以下のメソッドにpublic修飾子を付与します。
 - levelUpメソッド
 - setWazaメソッド
 - getStatusメソッド
 - useWazaメソッド
 - damagedメソッド

< 演習（ゲーム作成 その3） >
「Monster3.java」を作成します。

(3) 下記に従ってgetter/setterを作成してください。

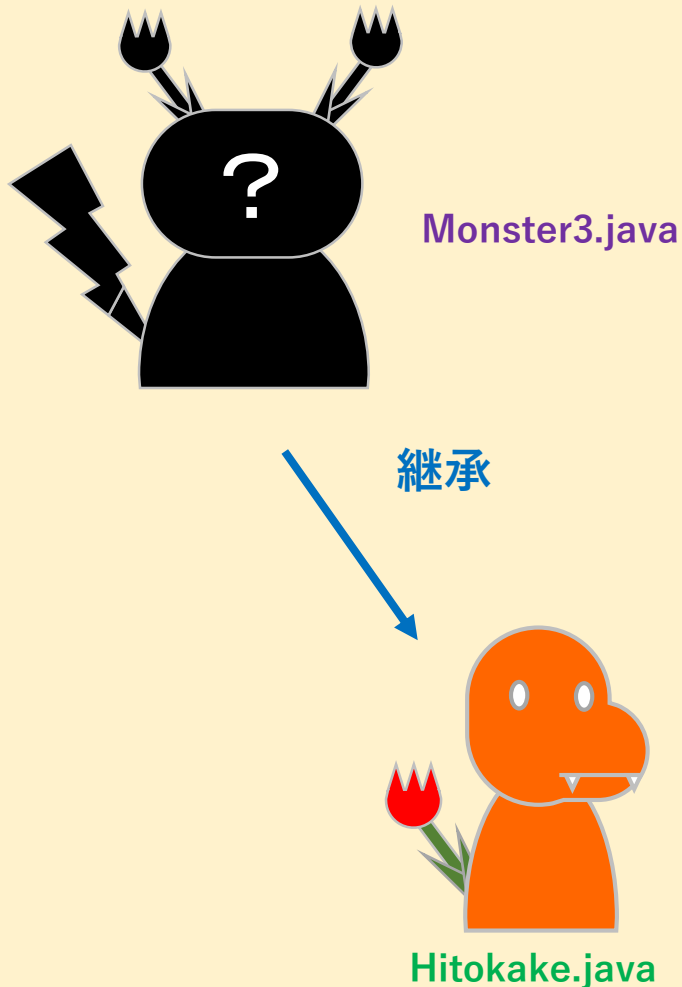


◆getter/setter

- ・ 以下のフィールドに対応するgetter/setterを作成します。
 - character
 - trainer
 - name
 - lv
 - hp
 - atk
 - def
 - spd
 - hpMax
 - wazaNm
 - wazaDmgRate

<演習（ゲーム作成 その3）>

Monster3.java を継承してサブクラス「Hitokake.java」を作成してください。



[コンストラクタ1（引数なし）]

- ・スーパークラスのコンストラクタ1（引数なし）を利用する（未定義でも可）
- ・スーパークラスのフィールドcharacterに文字列「ヒトカケ」を代入する

[コンストラクタ2（引数2つ：トレーナー,名前）]

- ・スーパークラスのコンストラクタ2を利用する
- ・スーパークラスのフィールドcharacterに文字列「ヒトカケ」を代入する

[コンストラクタ3（引数3つ：トレーナー,名前,初期レベル）]

- ・スーパークラスのコンストラクタ3を利用する
- ・スーパークラスのフィールドcharacterに文字列「ヒトカケ」を代入する

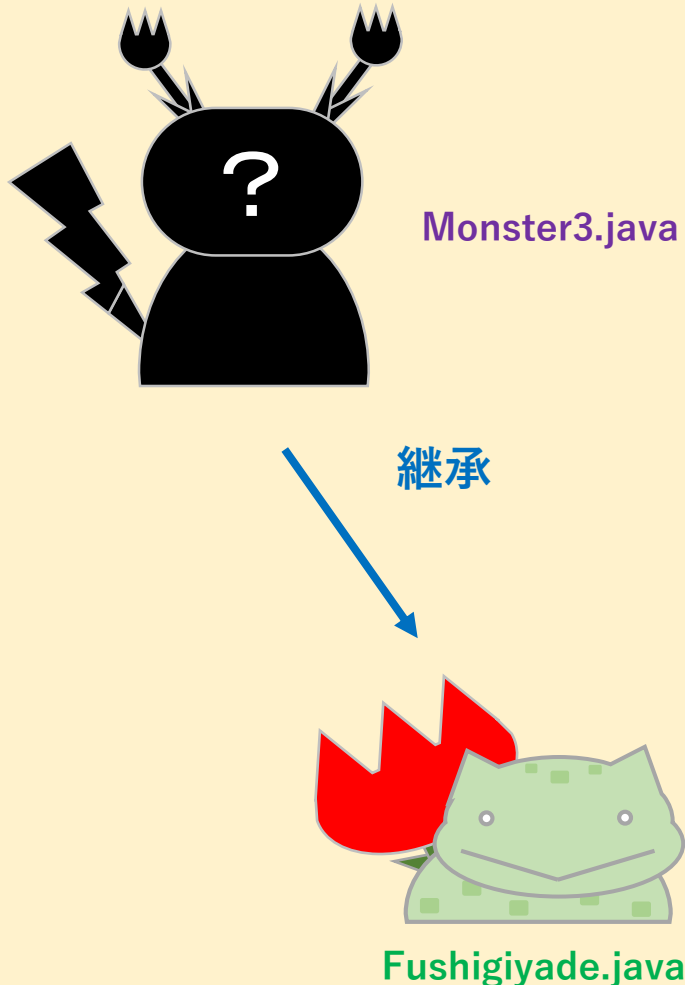
[levelUpメソッド（オーバーライド）]

<機能詳細>

- ・引数を元に一部スーパークラスのフィールドの値を更新します。
 - lv : 上昇レベル×1の値を既存値にプラスする
 - hp : 上昇レベル×29の値を既存値にプラスする
 - atk : 上昇レベル×8の値を既存値にプラスする
 - def : 上昇レベル×5の値を既存値にプラスする
 - spd : 上昇レベル×9の値を既存値にプラスする
 - hpMax : 更新後のhpの値を代入する

<演習（ゲーム作成 その3）>

Monster3.java を継承してサブクラス「Fushigiyade.java」を作成してください。



[コンストラクタ1（引数なし）]

- ・スーパークラスのコンストラクタ1（引数なし）を利用する（未定義でも可）
- ・スーパークラスのフィールドcharacterに文字列「フシギヤデ」を代入する

[コンストラクタ2（引数2つ：トレーナー,名前）]

- ・スーパークラスのコンストラクタ2を利用する
- ・スーパークラスのフィールドcharacterに文字列「フシギヤデ」を代入する

[コンストラクタ3（引数3つ：トレーナー,名前,初期レベル）]

- ・スーパークラスのコンストラクタ3を利用する
- ・スーパークラスのフィールドcharacterに文字列「フシギヤデ」を代入する

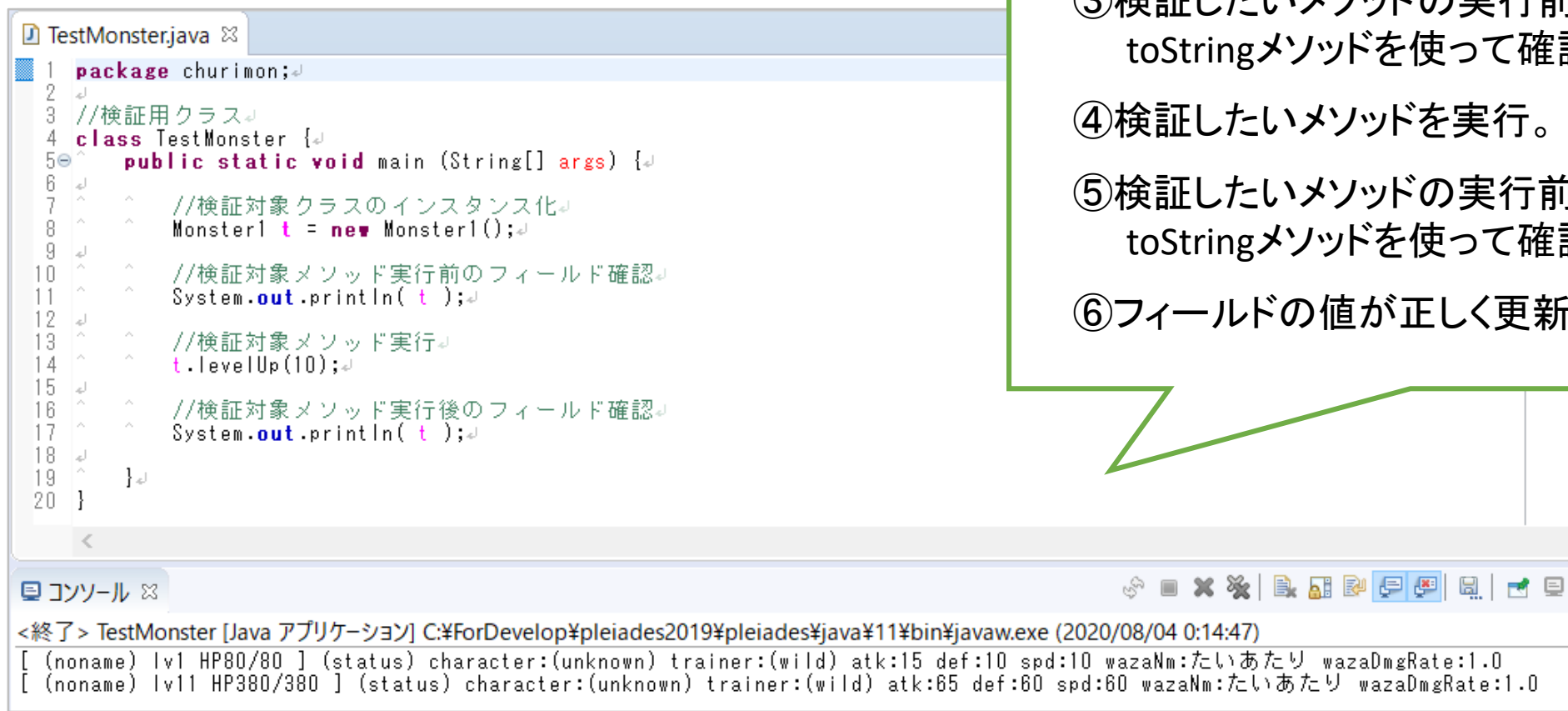
[levelUpメソッド（オーバーライド）]

<機能詳細>

- ・引数を元に一部スーパークラスのフィールドの値を更新します。
 - lv : 上昇レベル×1の値を既存値にプラスする
 - hp : 上昇レベル×31の値を既存値にプラスする
 - atk : 上昇レベル×6の値を既存値にプラスする
 - def : 上昇レベル×7の値を既存値にプラスする
 - spd : 上昇レベル×8の値を既存値にプラスする
 - hpMax : 更新後のhpの値を代入する

～ 作成したメソッドの検証方法 ～

- ①検証用クラス(TestMonster)を準備。
- ②検証したいクラスをインスタンス化。
- ③検証したいメソッドの実行前のフィールド状況をtoStringメソッドを使って確認。
- ④検証したいメソッドを実行。
- ⑤検証したいメソッドの実行前のフィールド状況をtoStringメソッドを使って確認。
- ⑥フィールドの値が正しく更新されているかを確認。



The screenshot shows a Java IDE with a file named `TestMonster.java` open. The code defines a package `churimon` and a class `TestMonster` with a `main` method. The `main` method performs the following steps: 1. Imports `Monster1`. 2. Creates an instance of `Monster1` named `t`. 3. Prints the state of `t` before calling `levelUp(10)`. 4. Calls `t.levelUp(10)`. 5. Prints the state of `t` after the method call. The console output shows the state of `t` before and after the `levelUp` method is called, confirming that the HP and other stats are updated correctly.

```
1 package churimon;
2
3 //検証用クラス
4 class TestMonster {
5     public static void main (String[] args) {
6
7         //検証対象クラスのインスタンス化
8         Monster1 t = new Monster1();
9
10        //検証対象メソッド実行前のフィールド確認
11        System.out.println(t);
12
13        //検証対象メソッド実行
14        t.levelUp(10);
15
16        //検証対象メソッド実行後のフィールド確認
17        System.out.println(t);
18    }
19 }
20 }
```

コンソール

<終了> TestMonster [Java アプリケーション] C:\ForDevelop\pleiades2019\pleiades\java\11\bin\javaw.exe (2020/08/04 0:14:47)

```
[ (noname) lvl HP80/80 ] (status) character:(unknown) trainer:(wild) atk:15 def:10 spd:10 wazaNm:たいあたり wazaDmgRate:1.0
[ (noname) lvl1 HP380/380 ] (status) character:(unknown) trainer:(wild) atk:65 def:60 spd:60 wazaNm:たいあたり wazaDmgRate:1.0
```