

# Java プログラミング基礎

例題集

### 例題 1 変数、リテラル、演算子

- 変数、定数の宣言、計算等を行う

実行結果

```
aとbのビット論理積は8<
a+=2は12<
1000円にかかる税額は80.0<
```

### 例題 2 配列

- int 型配列と String 型配列を用意し、値の格納と表示を確認する

実行結果

```
配列su[0]は10<
配列str[0]はA<
配列su[1]は20<
配列str[1]はB<
配列su[2]は30<
配列str[2]はC<
配列su[3]は40<
配列su[4]は50<
```

### 例題 3 if-2 分岐処理

- キーボードから入力した値が負の数であれば「負の数です」と表示してプログラムを終了する
- さらに、正の数で偶数なら「偶数です」、奇数ならば「奇数です」と表示する
- 今回の例では、0は正の数、偶数と同じ振る舞いとして処理しています

実行結果

数値を入力 -> -10< 負の数です<	数値を入力 -> 50< 偶数です<
-------------------------	-----------------------

#### 例題4 if-多分岐処理

- キーボードから入力した値が0であれば「0です」、負の数であれば「負の数です」と表示してプログラムを終了する
- さらに、正の数で偶数なら「偶数です」、奇数ならば「奇数です」と表示する

実行結果

```
数値を入力 -> 0< 0です<
数値を入力 -> -10< 負の数です<
数値を入力 -> 50< 偶数です<
```

#### 例題5 switch文

- キーボードから入力した値が0、または負の数であれば「正の数を入力してください」と表示してプログラムを終了する
- さらに、正の数で偶数なら「偶数です」、奇数ならば「奇数です」と表示する

実行結果

```
数値を入力 -> -1<
正の数を入力してください<
```

#### 例題6 繰り返し文

- 掛け算九九のプログラムを作成する

実行結果

```
while文による掛け算九九表示<
1の段: 1 2 3 4 5 6 7 8 9 <
2の段: 2 4 6 8 10 12 14 16 18 <
3の段: 3 6 9 12 15 18 21 24 27 <
4の段: 4 8 12 16 20 24 28 32 36 <
5の段: 5 10 15 20 25 30 35 40 45 <
6の段: 6 12 18 24 30 36 42 48 54 <
7の段: 7 14 21 28 35 42 49 56 63 <
8の段: 8 16 24 32 40 48 56 64 72 <
9の段: 9 18 27 36 45 54 63 72 81 <
```

### 例題 7 繰り返し文

- キーボードから値を 10 個入力し、int 型配列 su に格納する
- その後配列に格納された値の合計を求める

実行結果

```
値の入力: 1<
値の入力: 2<
値の入力: 3<
値の入力: 4<
値の入力: 5<
値の入力: 6<
値の入力: 7<
値の入力: 8<
値の入力: 9<
値の入力: 10<
合計点は 55<
```

### 例題 8 繰り返し文

- キーボードから値を 10 個入力し、int 型配列 su に格納する
- その後配列に格納された値の合計を求める

実行結果

```
値の入力: 1<
値の入力: 2<
値の入力: 3<
値の入力: 4<
値の入力: 5<
値の入力: 6<
値の入力: 7<
値の入力: 8<
値の入力: 9<
値の入力: 10<
合計点は 55<
```

### 例題 9 break, continue

- キーボードから値を 10 個入力し、int 型配列 su に格納する
- ただし、
  - 入力した値が 0 ならばキー入力を終了する
  - 入力した値が負の数であれば、配列 su には格納しないでキーボード入力を継続する
  - 結果として、0 が途中で入力されると値が 10 個分入らない可能性がある。また負の数が入力されると無視されるので、正の数が 10 個入るまでキーボード入力が繰り返される
- 配列に格納された値の合計を表示する

実行結果

```
値の入力: 1<
値の入力: -10<
値の入力: 2<
値の入力: 8<
値の入力: 9<
値の入力: 10<
値の個数は 10, 合計点は 55<
```

負の数は無視

### 例題 10 クラスの作成

- Sensor クラスの定義をする

データ型	名前	説明	設定値 (例)
String	productName	製品名	POLY35
String	manufacture	製造者名	ポリテケ
int	value	値	0

名前	説明	戻り値	引数
get	センサの値を取得	int	なし

### 例題 11 クラスのインスタンス化とメンバの利用

- 例題 10 で作成したクラス Sensor のインスタンスを生成して利用する

実行結果

製造者名: ポリテケ

製品名: POLY35

取得値: 0

名前 TestSensor

## 例題 12 コンストラクタ

- 例題 10 で作成した `Sensor` クラスを利用して暗黙のコンストラクタの確認をする  
例題 11 とソースコードは同じ

## 例題 13 コンストラクタ

- 例題 10 で作成した `Sensor` クラスに以下のコンストラクタを追加し、確認する
  - 引数を 2 つ取るコンストラクタ

引数	振る舞い
<code>String productName</code>	引数 <code>productName</code> の値をフィールド <code>product</code> に代入する

- 引数を 1 つ取るコンストラクタ

引数	振る舞い
<code>String productName</code>	引数 <code>productName</code> の値をフィールド <code>product</code> に代入する
<code>String manufacture</code>	引数 <code>manufacture</code> の値をフィールド <code>manufacture</code> に代入する

実行結果

No.1

製造者名: ポリテケ

製品名: POLY35

値: 0

No.2

製造者名: ポリテケ

製品名: POLY36

値: 0

No.3

製造者名: ポリテコ

製品名: POLY36

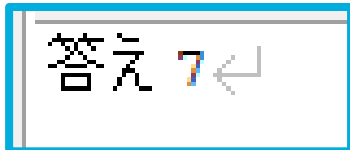
値: 0

#### 例題 14 インスタンスメンバ

- 以下のメンバを持つクラス Calc を作成し、インスタンス化して結果を表示する

クラス名	メンバの種類	データ型 (戻り値の型)	名前	値や振る舞い
Calc	フィールド	int	answer	加算結果がセットされる
	メソッド	void	add(int x,int y)	引数 x、y の合計を求める

実行結果

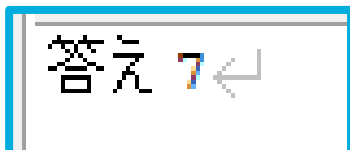
A screenshot of a terminal window with a blue border. Inside, the text '答え 7' is displayed in a monospaced font, followed by a cursor character '←'.

#### 例題 15 クラスメンバ

- 以下のメンバを持つクラス Calc を作成し、結果を表示する

クラス名	メンバの種類	データ型 (戻り値の型)	名前	値や振る舞い
Calc	フィールド	int	answer	加算結果がセットされる
	メソッド	void	add(int x,int y)	引数 x、y の合計を求める

実行結果

A screenshot of a terminal window with a blue border. Inside, the text '答え 7' is displayed in a monospaced font, followed by a cursor character '←'.

## 例題 16 アクセスメソッド

- キーボードから入力した、氏名と点数情報を以下の仕様を持つクラス Student にアクセスメソッドを利用して設定取得を行う

クラス名	メンバの種類	データ型 (戻り値の型)	名前	値
Student	フィールド	String	name	初期値なし
	フィールド	int	tensu	初期値なし
	メソッド	void	setName(String name)	
	振る舞い	引数 name の値をフィールド name に設定		
	メソッド	String	getName()	
	振る舞い	フィールド name の値を取得		
	メソッド	void	setTensu(int tensu)	
	振る舞い	引数 tensu の値をフィールド tensu に設定		
	メソッド	String	getTensu()	
	振る舞い	フィールド tensu の値を取得		

実行結果

※アクセスメソッドの追加手順 (Eclipse)

- ① 新規で Student.java を作成する
- ② クラスのメンバに private でフィールドを宣言する

```
1. public class Student {
2.     private String name;    // 名前格納用
3.     private int tensu;      // 点数格納用
4. }
```

```
名前の入力:Test<
点数の入力:85<
入力した情報は...<
名前: Test<
点数: 85<
```

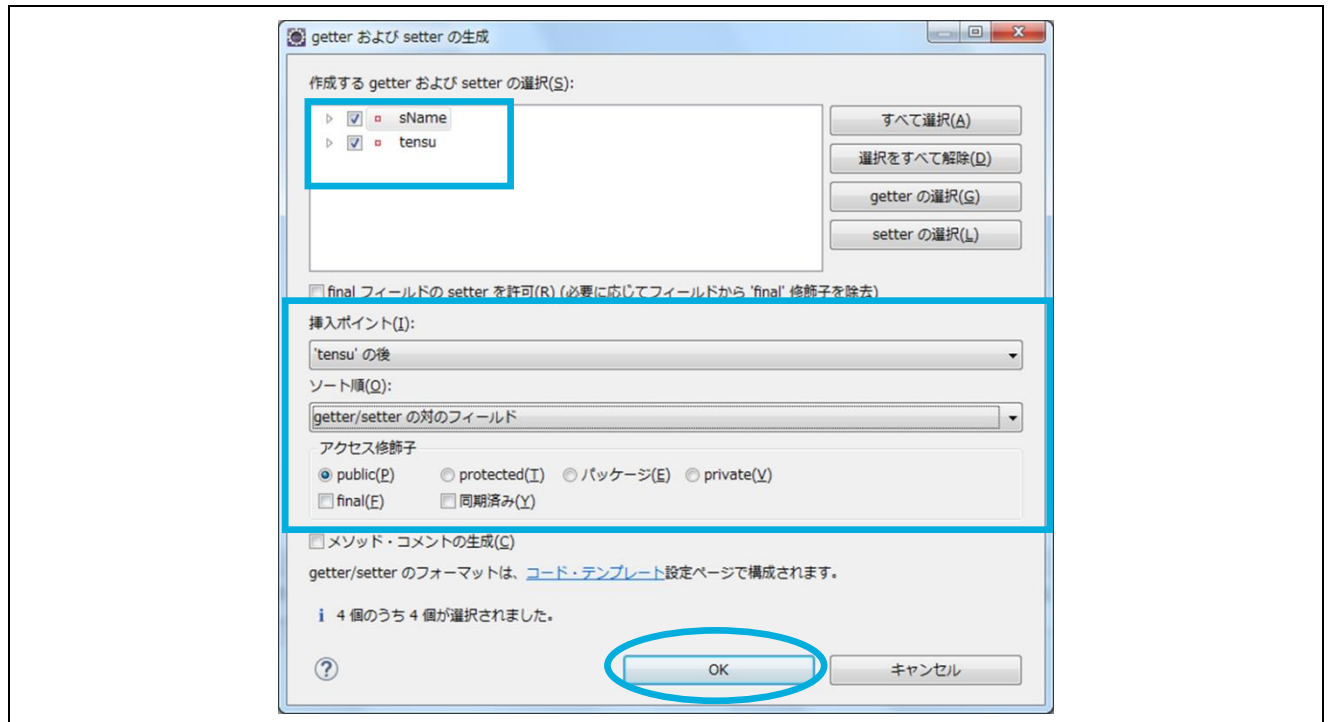
- ③ Eclipse のメニュー「ソース」→「getter および setter の生成」を選択。(この時、Eclipse のエディタ画面の対象となるクラス内でカーソルが点滅していること)



- ④ Getter/Setter メソッドを作成したいフィールドに ☒ を入れ、メソッドの挿入位置、並び替え、



アクセス制御を設定し、OK を押す



- ⑤ Eclipse では、フィールドだけ入力すれば、機能を使ってアクセスメソッドを自動挿入することができる

### 例題 17 継承

- 以下の仕様を持つクラス `temperatureSensor` を作成する
- ただし、`TemperatureSensor` は `Sensor` クラスを継承する
- コンストラクタ

引数	振る舞い
<code>String productName</code>	引数 <code>productName</code> の値を親のフィールド <code>productName</code> に代入する
<code>String manufacture</code>	引数 <code>manufacture</code> の値を親のフィールド <code>manufacture</code> に代入する

- フィールド

名前	型	値
<code>temperature</code>	<code>int</code>	温度を格納する

- メソッド

名前	戻り値	引数	振る舞い
<code>get</code>	<code>int</code>	なし	フィールド <code>temperature</code> を返す

### 例題 18 アップキャストとダウンキャスト

- 例題 17 の `TemperatureSensor` クラスの仕様にメソッドを追加し、アップキャストとダウンキャストを確認する

名前	戻り値	引数	振る舞い
<code>getFahrenheit</code>	<code>int</code>	なし	華氏を返す
<code>getCelsius</code>	<code>Int</code>	なし	摂氏を返す

### 例題 19 オーバーロード

- 例題 18 の `TemperatureSensor` クラスの仕様にメソッドを追加して、オーバーロードの確認をする

戻り値	名前	説明
<code>int</code>	<code>get(int unit)</code>	引数の種類に依り、摂氏と華氏を返す
<code>unit</code> が <code>TemperatureSensor.Celsius</code> のときは摂氏を返す		
<code>unit</code> が <code>TemperatureSensor.Fahrenheit</code> のときは華氏を返す		

### 例題 20 オーバーライド

- 例題 18 の仕様にメソッドを追加して、オーバーライドの確認をする

クラス名	戻り値	名前
<code>Sensor</code>	<code>String</code>	<code>toString()</code>
	クラス名、製造者と製品名を含んだ文字列を生成	
<code>TemperatureSensor</code>	<code>String</code>	<code>toString()</code>
	クラス名、製造者と製品名を含んだ文字列を生成	

## 例題 21 抽象クラス

- Sensor クラスを抽象クラスとし、以下の抽象メソッドを追加する。

クラス名 又は インターフェース名	メンバの種類	データ型 (戻り値の型)	名前	値
Shape (インターフェース)	定数	double	PI	3.141592
	抽象メソッド	double	getArea()	
Rect (Shape を実装)	フィールド	double	width	
	フィールド	double	height	
	コンストラクタ	なし	Rect(double width,double height)	
	振る舞い	引数 width,height の値をフィールドに代入		
	メソッド (オーバーライド)	double	getArea()	
	振る舞い	width と height の幅,高さを持つ四角形の面積を求める		
Circle (Shape を実装)	フィールド	double	radius	
	コンストラクタ	なし	Circle(double radius)	
	振る舞い	引数 radius の値をフィールドに代入		

戻り値	名前
int	get()

## 例題 22 インターフェース

- 以下の仕様を持つクラスやインターフェースを定義し、動作確認をする

メソッド (オーバーライド)	double	getArea()
----------------	--------	-----------

振る舞い	半径が <code>radius</code> の円の面積を求める
------	-----------------------------------

## 例題 23 例外処理

- キーボードから2つの数値を入力して、除算した結果を表示する。
- ただし、以下の条件のとき例外を発生させて対応する

条件	例外名	対応処理
数値でない値をキーボードから入力した	InputMismatchException	数値を入力してくださいとメッセージを表示し終了
0 で除算してしまった	ArithmeticException	0 で割っていますとメッセージを表示し終了

実行結果

実行結果

```

数値を入力 -> g↵
数値を入力してください↵
java.util.InputMismatchException↵
> at java.util.Scanner.throwFor(Scanner.java:864)↵
> at java.util.Scanner.next(Scanner.java:1485)↵
> at java.util.Scanner.nextDouble(Scanner.java:2413)↵
> at reidai25.Reidai25.main(Reidai25.java:15)↵

```

```

数値を入力 -> 10↵
数値を入力 -> 0↵
0で割っています↵
java.lang.ArithmeticException: / by zero↵
> at reidai25.Reidai25.main(Reidai25.java:25)↵

```

```

数値を入力 -> 10↵
数値を入力 -> 2↵
結果は... 5.0↵

```

## 例題 24 匿名クラス

- 以下のような仕様を持つクラスを作成し、動作確認をする

クラス名	メンバの種類	データ型 (戻り値の型)	名前	値
Shape	フィールド	double	width	
	フィールド	double	height	
	メソッド	void	getArea()	
	振る舞い	width と height の幅, 高さを持つ四角形の面積を求めて表示		
匿名クラス内 (Shape クラスのサブクラス)	フィールド	double	radius	3.0
	メソッド	void	getArea()	
	振る舞い	半径 radius の円の面積を求めて表示		

実行結果

```

<終了> Reidai24 [Java アプリケーション] C:\pleiade
半径3.0の円の面積は: 28.274328↵
10×5の四角形の面積は: 50.0↵

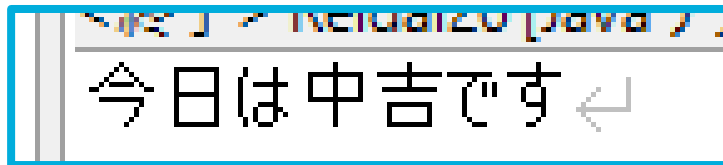
```

## 例題 25 列挙型

- 以下の仕様を持つ列挙型を生成し、おみくじアプリケーションを作成する

項目	設定値
列挙型名	Omikuji
列挙子リスト	DAIKICHI,CHUKICHI,KICHI,KYO,DAIKYO

実行結果



## 例題 26 コレクションクラス

- クラス、コレクションクラスを作成して、データを格納して動作を確認する

項目	設定値
クラス名	City
フィールド	String nameofcity
フィールド	double longitude
フィールド	double latitude

City 型インスタンス city に格納するデータ

項目	設定値
city[0].nameofcity	東京
city [0].longitude	139.766084
city [0].latitude	35.681382
city[1].nameofcity	ロンドン
city[1].longitude	-0.127758
City[1].latitude	51.507351

City 型 ArrayList のインスタンス array に追加するデータ

city[0]	city[1]
---------	---------

キーString, 値 City 型 HashMap のインスタンス hash に追加するデータ

(キー, 値)=(“Tokyo”,city[0])	(キー, 値)=(“London”,city[1])
---------------------------	----------------------------

表示する項目

ArrayList インスタンスに格納した City クラスの nameofcity フィールド情報
HashMap インスタンスに格納したキー「Tokyo」の値 nameofcity フィールド情報

実行結果

<終了> TestException [Java

ArrayList [0]=東京

ArrayList [1]=ロンドン

HashMap Tokyo=東京