

pi-2. Java プログラミングの基本

トピックス：オブジェクトとメソッド，引数，代入，データの種類，制御（Java Tutor による演習）

URL: <https://www.kkaneko.jp/cc/pi/index.html>

（Java の基本）

金子邦彦



オブジェクト, メソッド, 代入, 変数



- **オブジェクト** : コンピュータでの 操作や処理の対象となるもの のこと
- **メソッド** : **オブジェクト** に属する操作や処理. **メソッド** 呼び出しでは, **引数** を指定することがある. **引数** (ひきすう) は, **メソッド** に渡す値のこと

Hero.attack("fence", 36, 26)

- **代入** : 「=」を使用. オブジェクトの 値が変化 する

b = a + 100

- **データの種類**

int x = 100 . . . 整数

String s = "abc" . . . 文字列

- **変数** : 名前の付いたオブジェクト には, **変数**, **関数** などがある (「変数」は, 数学の変数とは違う意味)

番号	項目
	復習
2-1	オブジェクトとメソッド, 引数, 代入
2-2	データの種類
2-3	制御

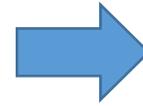
各自、資料を読み返したり、課題に取り組んだりも行う

この授業では、**Java** を用いて基礎を学び、マスターする

プログラミング (programming)

- コンピュータは、プログラムで動く
- プログラミングは、プログラムを設計、製作すること
- 何らかの作業を、コンピュータで実行させるために行う

```
public class YourClassNameHere {  
    public static void main(String[] args) {  
        int x = 100;  
        int y = 200;  
        System.out.printf("%d", x * y);  
    }  
}
```



20000

プログラムの
ソースコード
(Java 言語)

プログラムの
実行結果

ソースコード (source code)

- **プログラム**を, 何らかの**プログラミング言語**で書いたもの
- 「**ソフトウェアの設計図**」ということも.

人間も読み書き, 編集できる

```
public class YourClassNameHere {  
    public static void main(String[] args) {  
        int x = 100;  
        int y = 200;  
        System.out.println(x + y);  
    }  
}
```

100 × 200 を計算する Java 言語プログラム

プログラムが役に立つ理由



- ① プログラム次第で，様々な処理が可能.
- ② プログラムは，コンピュータでの様々な処理を自動化する
- ③ プログラムのソースコードは，作業記録としても使うことができる．いつでも再現できる．
- ④ プログラム中の値などを変えて再実行も簡単

Java Tutor の起動



① **ウェブブラウザ**を起動する

② **Java Tutor** を使いたいのので, 次の URL を開く
<http://www.pythontutor.com/>

③ 「**Java**」 をクリック ⇒ **編集画面**が開く

Learn Python, JavaScript, C, C++, and Java

This tool helps you learn Python, JavaScript, C, C++, and Java programming by [visualizing code execution](#). You can use it to debug your homework assignments and as a supplement to online coding tutorials.

Start coding now in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

Over 15 million people in more than 180 countries have used Python Tutor to visualize over 200 million pieces of code. It is the most widely-used program visualization tool for computing education.

You can also embed these visualizations into any webpage. Here's an example showing recursion in Python:

Java Tutor でのプログラム実行手順



```
1 public class YourClassNameHere {
2     public static void main(String[] args) {
3         int x = 100;
4         int y = 200;
5         System.out.printf("%d\n", x + y);
6     }
7 }
```

Visualize Execution



```
Java 8
(known limitations)

1 public class YourClassNameHere {
2     public static void main(String[] args) {
3         int x = 100;
4         int y = 200;
5         System.out.printf("%d\n", x + y);
6     }
7 }
```

Edit this code

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 1 of 5



(1) 「Visualize Execution」をクリックして実行画面に切り替える

(2) 「Last」をクリック。

Print output (drag lower right corner to resize)

300

Frames	Objects
main:6	
x	100
y	200
Return value	void



```
Java 8
(known limitations)

1 public class YourClassNameHere {
2     public static void main(String[] args) {
3         int x = 100;
4         int y = 200;
5         System.out.printf("%d\n", x + y);
6     }
7 }
```

Edit this code

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Done running (5 steps)

(3) 実行結果を確認する。

(4) 「Edit this code」をクリックして編集画面に戻る

Java Tutor 使用上の注意点①



- 実行画面で、次のような赤の表示が出ることもある →
無視してよい

過去の文法ミスに関する確認表示
邪魔なときは「Close」

Python Tutor: Visualize code in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

Java 8
([known limitations](#))

```
1 public class YourClassNameHere {
2     public static void main(String[] args) {
3         int x = 100;
4     }
5 }
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 1 of 3

[Customize visualization](#)

Frames Objects

main:3

You just fixed the following error:

```
1 public class YourClassNameHere {
2     public static void main(String[] args) {
3         int x = 100
4     }
5 }
```

Error: ';' expected

Please help us improve this tool with your feedback.
What misunderstanding do you think caused this error?

Submit **Close** [hide all of these pop-ups](#)

Java Tutor 使用上の注意点②



「please wait ... executing」のとき、10秒ほど待つ。

Python Tutor: Visualize code in [Python](#), [Ja](#)

Please wait ... your code is running (up to 10 seconds)

Write code in [Java 8](#)

```
1 public class YourClassNameHere {
2     public static void main(String[] args) {
3         int x = 100;
4     }
5 }
```

Please wait ... executing (takes up to 10 seconds)

- 混雑しているときは、「Server Busy・・・」
というメッセージが出ることがある。
混雑している。少し（数秒から数十秒）待つと自
動で表示が変わる（変わらない場合には、操作を
もう一度行ってみる）

2-1. オブジェクトとメソッド, 引数, 代入

オブジェクト



- **オブジェクト**：コンピュータでの**操作や処理の対象となるもの**のこと
- 名前の付いたオブジェクトには、**変数**、**関数**などがある。

オブジェクトとメソッド



hero.moveDown()

hero **オブジェクト**
moveDown() **メソッド**
間を「.」で区切っている

- **メソッド: オブジェクト**に属する操作や処理.
- **メソッド**呼び出しでは, **引数**を指定することがある. **引数** (ひきすう) は, **メソッド**に渡す値のこと

hero.attack("fence", 36, 26)

オブジェクトとメソッド



オブジェクトが動く



実行画面

```
1 # 宝石まで移動させよう！  
2 # 壁に当たったらダメだぞ！  
3 # 下にコードを打ち込め！  
4  
5 ▶ hero.moveRight()
```

オブジェクトとメソッド

オブジェクトが動く



実行画面

```
1 # 宝石まで移動させよう！  
2 # 壁に当たったらダメだぞ！  
3 # 下にコードを打ち込め！  
4  
✓ 5 hero.moveRight()  
▶ 6 hero.moveDown()
```

オブジェクトとメソッド

メソッドの引数



オブジェクトが動く



実行画面

```
1 # Use arguments with  
   farther.  
▶ 2 hero.moveRight(3)  
3  hero.moveUp()  
4  hero.moveRight()  
5  hero.moveDown(3)  
6  hero.moveRight(3)
```

オブジェクトとメソッド

引数がある場合もあれば、
ない場合もある。

代入



- **代入** : プログラムで, 「**x = 100**」 のように書くと, **x の値が 100 に変化** する

x = 100

プログラム



Frames

Global frame

x | 100

実行結果

Java プログラムの書き方



プログラムの例

```
x = 100
a = x + 200
enemy1 = hero.findNearestEnemy()
hero.attack(enemy1)
```

- **代入** : **オブジェクト名** + 「**=**」
+ 式または値またはメソッド呼び出し
- **メソッドアクセス** : **オブジェクト名** + 「**.**」
+ **メソッド名** + 「**()**」 (引数を付けることも)

その他, 属性アクセス, 関数呼び出し, 制御, 「*」, 「+」などの演算子, コマンド, 定義など

2-2. データの種類

Java のデータの種類



① 基本データ

データの種類	基本データ型	サイズ
整数	byte	8 bit
	short	16 bit
	int	32 bit
	long	64 bit
浮動小数	float	32 bit
	double	64 bit
文字	char	16 bit
true/false	boolean	

② 基本データの配列

③ クラスに属するオブジェクト: String クラスなど多種

演習

資料 : 22 ~ 25

【トピックス】

- データの種類
- 変数

変数

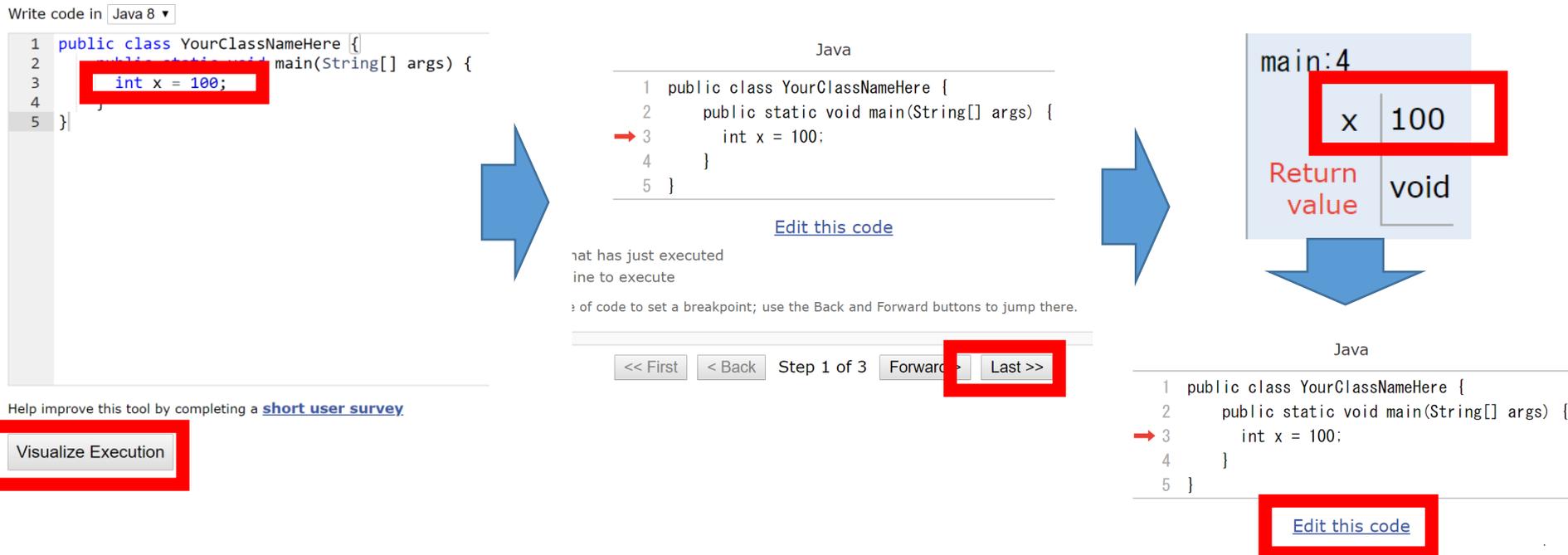


- ① Java Tutor のエディタで次のプログラムを入れる。
整数を使ってみる。
変数 x の値を 100 に変化させる。
次のように「`int x = 100;`」を入れる。

Write code in

```
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
3         int x = 100;  
4     }  
5 }
```

② 実行し，結果を確認する 「x 100」となっている。



Write code in Java 8

```
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
3         int x = 100;  
4     }  
5 }
```

Java

```
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
3         int x = 100;  
4     }  
5 }
```

[Edit this code](#)

mat has just executed
ine to execute

of code to set a breakpoint; use the Back and Forward buttons to jump there.

<< First < Back Step 1 of 3 Forward > Last >>

Frames

main:4

x	100
Return value	void

Java

```
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
3         int x = 100;  
4     }  
5 }
```

[Edit this code](#)

Help improve this tool by completing a [short user survey](#)

Visualize Execution

「**Visual Execution**」をクリック。そして「**Last**」をクリック。結果を確認。
「**Edit this code**」をクリックすると，エディタの画面に戻る

- ③ Java Tutor のエディタで次のプログラムを入れる。
今回は、**文字列**を使ってみる

```
1 public class YourClassNameHere {
2     public static void main(String[] args) {
3         int x = 100;
4         String s = "abc";
5     }
6 }
```

④ 実行し, 結果を確認する. 「s "abc"」となっている.

```
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
3         int x = 100;  
4         String s = "abc";  
5     }  
6 }
```

Visualize Execution



Java 8
([known limitations](#))

```
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
3         int x = 100;  
4         String s = "abc";  
5     }  
6 }
```

[Edit this code](#)

Step 1 of 4

Last >>



Frames

main:5	
x	100
s	"abc"
Return value	void

↓

Java 8
([known limitations](#))

```
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
3         int x = 100;  
4         String s = "abc";  
5     }  
6 }
```

[Edit this code](#)

「**Visual Execution**」をクリック. そして「**Last**」をクリック. 結果を確認.
「**Edit this code**」をクリックすると, エディタの画面に戻る

2-3. 制御

- プログラムは、上から順に実行（逐次実行）が基本である
- 条件分岐では、**「実行される部分」と「実行されない部分」**がある
- 繰り返し（ループ）では、**同じ部分が繰り返し実行**される

条件分岐



条件分岐では、「実行される部分」と「実行されない部分」がある

```
1 public class Main {
2     public static void main(String[] args) {
3         int age = 10;
4         if (age <= 12) {
5             System.out.printf("500 yen");
6         } else {
7             System.out.printf("1200 yen");
8         }
9     }
```

プログラム

age <= 12 のときのみ

System.out.printf("500 yen") が実行される

age > 12 のときのみ

System.out.printf("1200 yen") が実行される

繰り返し (ループ)

繰り返し (ループ) では, **同じ部分が繰り返し実行される**

```
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
3         int s = 0;  
4         for(int i = 0; i < 6; i++) {  
5             s = s + i;  
6         }  
7         System.out.println(s);  
8     }  
9 }
```



Print output (drag lower i

15

実行結果

プログラム

足し算の 5 回繰り返し

0 + 1, 1 + 2, 3 + 3, 6 + 4, 10 + 5

オブジェクト, メソッド, 代入, 変数



- **オブジェクト** : コンピュータでの 操作や処理の対象となるもの のこと
- **メソッド** : **オブジェクト** に属する操作や処理. **メソッド** 呼び出しでは, **引数** を指定することがある. **引数** (ひきすう) は, **メソッド** に渡す値のこと

Hero.attack("fence", 36, 26)

- **代入** : 「=」を使用. オブジェクトの 値が変化 する

b = a + 100

- **データの種類**

int x = 100 . . . 整数

String s = "abc" . . . 文字列

- **変数** : 名前の付いたオブジェクト には, **変数**, **関数** などがある (「変数」は, 数学の変数とは違う意味)

関連ページ

- **Java プログラミング入門**

GDB online を使用

<https://www.kkaneko.jp/cc/ji/index.html>

- **Java の基本**

Java Tutor, GDB online を使用

<https://www.kkaneko.jp/cc/pi/index.html>

- **Java プログラム例**

<https://www.kkaneko.jp/pro/java/index.html>