

ji-4. Javaプログラミングにおける基本データ型と数値計算の基礎

(Java プログラミング入門)

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



金子邦彦



- 例題 1 . 硬貨の金種計算
- 例題 2 . うるう年の判定

整数の変数

浮動小数点数と整数の違い

- 例題 3 . 複利計算

整数の変数と、浮動小数点数の変数を混在させるときに気を付けねばならないこと

- プログラムでの**整数と浮動小数点数**の違いについて理解する
- 目的に応じて、**整数の変数**，**浮動小数点数の変数**を**正しく使い分ける**ことができるようになる

Java のデータの種類



・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
浮動小数点数	float	32 bit
	double	64 bit
文字	char	16 bit
true/false	boolean	

- ・基本データの**配列**
- ・**クラス**に属する**オブジェクト**: **String** クラスなど多種

この資料では、整数 **int** と浮動小数点数 **double** を使う4

整数と浮動小数点数



• 整数

整数（正か負か0）

例)

0

3

2 8

4 7 7 8

− 1

− 1 0

− 1 2 5 0

• 浮動小数点数

小数付きの数も可

例)

0

3

1 2 7 8 7 4 8 6 2 3

− 4 5 6 3 7 5 9 3 9

8

2. 1 9 0 8 7 2

0. 0 0 0 1 7 8

整数と浮動小数点数



	整数	浮動小数点数
四則演算	$+$, $-$, $*$, $/$ ($/$ では, 余りは切り捨て)	$+$, $-$, $*$, $/$
剰余	$\%$	
小数点以下切り捨て		floor

- プログラミングを行えるオンラインのサービス

<https://www.onlinegdb.com>

- ウェブブラウザを使う

- たくさんの言語を扱うことができる

Python3, Java, C/C++, C#, JavaScript,
R, アセンブリ言語, SQL など

- オンラインなので、「秘密にしたいプログラム」
を扱うには十分な注意が必要

Online GDB で Java を動かす手順



① ウェブブラウザを起動する

② 次の URL を開く

<https://www.onlinegdb.com>

A screenshot of a web browser's address bar. It features a magnifying glass icon on the left, followed by the text "https://www.onlinegdb.com". The address bar is set against a light gray background.

③ 「Language」 のところで, 「Java」 を選ぶ

SPONSOR Slack — Bring your team together with Slack, the collaboration hub for work.

Run Debug Stop Share Save Beautify

Language -- select --

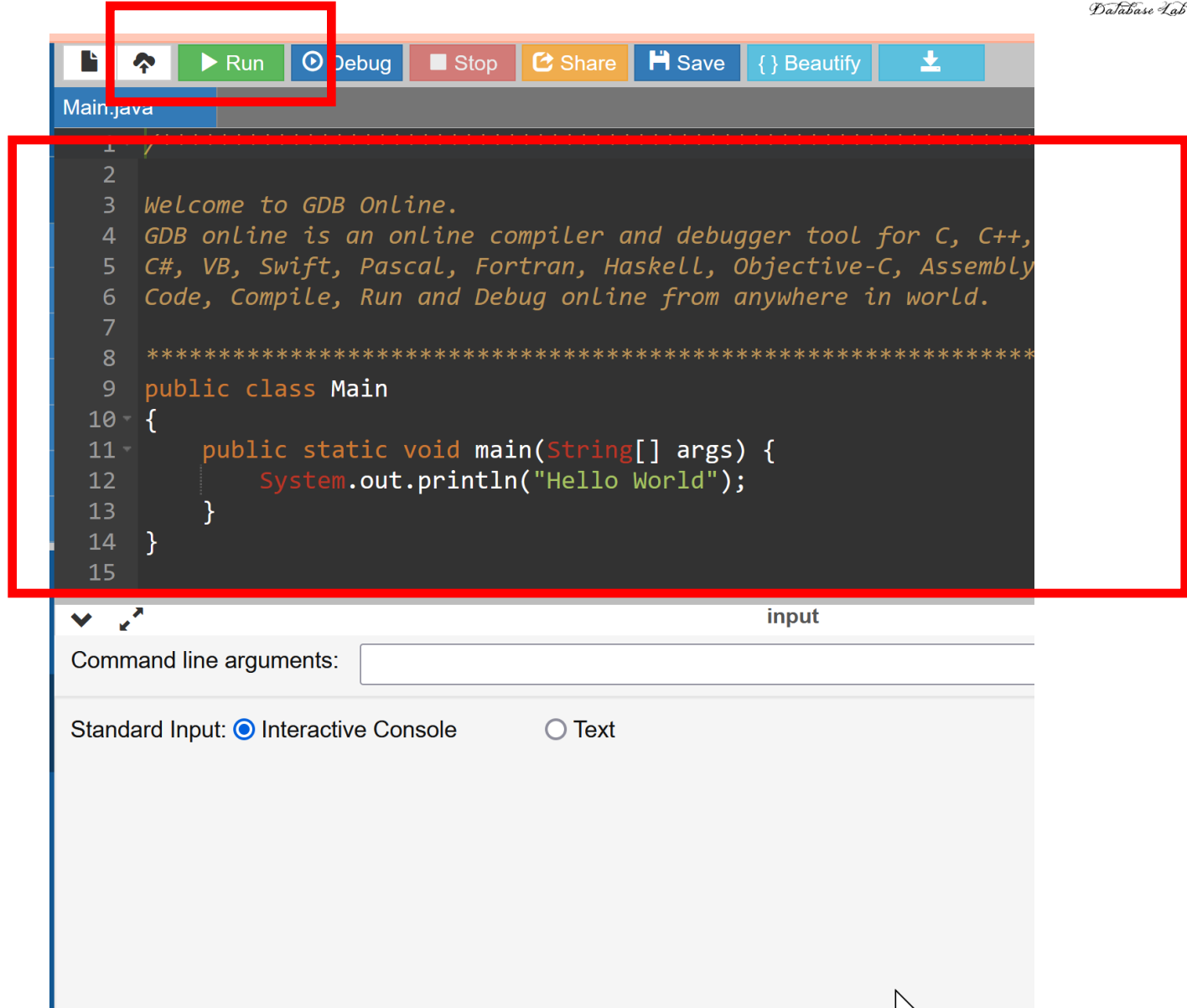
source code

```
1 /*****  
2  
3 Welcome to GDB Online.  
4 GDB online is an online compiler and debugger tool for C, C++, Python  
5 C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 *****/  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17
```

-- select --

- C
- C++
- C++ 14
- C++ 17
- Java**
- Python 3
- PHP
- C#
- VB
- HTML,JS,CSS
- Ruby
- Perl
- Pascal
- R
- Fortran
- Haskell
- Assembly(GCC)
- Objective C
- SQLite

実行ボタン



The screenshot shows the GDB Online IDE interface. The top toolbar contains buttons for Run, Debug, Stop, Share, Save, Beautify, and Download. The 'Run' button is highlighted with a red box. Below the toolbar is the code editor for 'Main.java', which contains the following code:

```
1 //
2
3 Welcome to GDB Online.
4 GDB online is an online compiler and debugger tool for C, C++,
5 C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly
6 Code, Compile, Run and Debug online from anywhere in world.
7
8 *****
9 public class Main
10 {
11     public static void main(String[] args) {
12         System.out.println("Hello World");
13     }
14 }
15
```

Below the code editor is the 'input' panel. It includes a 'Command line arguments:' field and a 'Standard Input:' section with two radio buttons: 'Interactive Console' (selected) and 'Text'.

エディタ画面

プログラムを
書き換えること
ができる

例題 1 . 硬貨の金種計算



- **金額を読み込んで，適切な小銭の枚数を求め，表示するプログラムを作る．**

例） 金額が 7 6 8 円 のとき，

5 0 0 円 玉 ： 1 枚

1 0 0 円 玉 ： 2 枚

5 0 円 玉 ： 1 枚

1 0 円 玉 ： 1 枚

5 円 玉 ： 1 枚

1 円 玉 ： 3 枚

- 例題では，簡単のため，**紙幣は考えない（硬貨のみ）**ということにする
- 各硬貨の枚数を扱うために，**整数の変数**を使う

- 1 円札の枚数
金額を 5 で割った余り
- 5 円玉の枚数
金額を 10 で割った余り（小数点以下切り捨て）について、
それを 5 で割った商
- 1 0 円玉の枚数
金額を 50 で割った余り（小数点以下切り捨て）について、
それを 10 で割った商

他の効果も同様に考える

```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
        int kingaku, n500, n100, n50, n10, n5, n1;
        Scanner s = new Scanner(System.in);
        System.out.println("Please Enter kingaku =");
```

```
kingaku = s.nextInt();
```

```
n500 = kingaku / 500;
n100 = ( kingaku % 500 ) / 100;
n50 = ( kingaku % 100 ) / 50;
n10 = ( kingaku % 50 ) / 10;
n5 = ( kingaku % 10 ) / 5;
n1 = kingaku % 5;
```

```
System.out.printf("500 en %d mai¥n", n500);
System.out.printf("100 en %d mai¥n", n100);
System.out.printf("50 en  %d mai¥n", n50);
System.out.printf("10 en  %d mai¥n", n10);
System.out.printf("5 en   %d mai¥n", n5);
System.out.printf("1 en   %d mai¥n", n1);
```

キーボードからの
データの読み込み

計算

画面表示

実行結果の例

```
Please Enter kingaku =  
12687  
500 en 25 mai  
100 en 1 mai  
50 en 1 mai  
10 en 3 mai  
5 en 1 mai  
1 en 2 mai
```

プログラム実行順



メッセージを表示

```
System.out.println("Please Enter kingaku =");
```

```
kingaku = s.nextInt();
```

整数データを読み込み

```
n500 = kingaku / 500;  
n100 = ( kingaku % 500 ) / 100;  
n50 = ( kingaku % 100 ) / 50;  
n10 = ( kingaku % 50 ) / 10;  
n5 = ( kingaku % 10 ) / 5;  
n1 = kingaku % 5;;
```

金種計算

```
System.out.printf("500 en %d mai¥n", n500);  
System.out.printf("100 en %d mai¥n", n100);  
System.out.printf("50 en  %d mai¥n", n50);  
System.out.printf("10 en  %d mai¥n", n10);  
System.out.printf("5 en   %d mai¥n", n5);  
System.out.printf("1 en   %d mai¥n", n1);
```

計算結果を表示

例題 2. うるう年の判定



- 「西暦年」を読み込んで、うるう年かどうか表示するプログラムを作る.

2022 → うるう年でない

2024 → うるう年である

- うるう年の判定のために、**比較演算**と**論理演算**を組み合わせる
- 西暦年が 4, 100, 400 の倍数であるかを調べるために % を使う (余りが 0 ならば, 倍数である)

グレゴリオ暦でのうるう年



- **うるう年とは： 2月が29日までである年**
 - うるう年は400年に97回で，1年の平均日数は365.2422日
 - **うるう年の判定法**
 - **年数が4の倍数の年 → うるう年**
 - **但し，100の倍数の年で400の倍数でない年**
 - **うるう年ではない（4の倍数だが例外とする）**
- (例) 2024年：うるう年（4の倍数）
2020年：うるう年（4の倍数）
2000年：うるう年（4の倍数）
1900年：**うるう年ではない**
(100の倍数だが400の倍数でない)
1800年：**うるう年ではない**
(100の倍数だが400の倍数でない)

```
import java.util.Scanner;
```

```
public class Main
```

```
{
```

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

```
        int y;
```

```
        Scanner s = new Scanner(System.in);
```

```
        System.out.println("Please Enter year =");
```

```
        y = s.nextInt();
```

条件式

```
        if (((y % 400) == 0) || (((y % 100) != 0) && ((y % 4) == 0))) {
```

```
            System.out.printf("%d is leap year¥n", y);
```

**条件が成り立つ場合
に実行される部分**

```
        } else {
```

```
            System.out.printf("%d is not leap year¥n", y);
```

```
        }
```

**条件が成り立たない場合
に実行される部分**

```
    }
```

```
}
```

実行結果の例

```
Please Enter year =  
2024  
2024 is leap year
```

```
Please Enter year =  
2000  
2000 is leap year
```

```
Please Enter year =  
1900  
1900 is not leap year
```

うるう年の判定式



$((y \% 400) == 0) \parallel (((y \% 100) != 0) \&\& ((y \% 4) == 0))$

400の倍数である

100の倍数でない

4の倍数である

または

かつ

例題 3 . 複利計算



- **元金**をある**年利**で，ある**年数**だけ運用したときの**元利**と**利息**を表示する
 - 複利計算では，**利息が利息**を生む.
 - 複利計算を行うために，**pow メソッド** ($\text{pow}(x,y)$ は， x の y 乗) を使う
 - 浮動小数点数の**小数点以下切り捨て**のために **floor メソッド** を使う
 - **整数**と**浮動小数点数**のデータが混在する
 - 元金 gankin: 整数データ (単位は 円)
 - 年数 nensu: 整数データ (単位は 年)
 - 年利 nenri: 浮動小数点数データ (単位は %)

```
import java.lang.Math;
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
        int gankin, nensu, ganri;
        double nenri, r;
        Scanner s = new Scanner(System.in);
        System.out.println("Please Enter gankin (en) =");
        gankin = s.nextInt();
        System.out.println("Please Enter nensu (en) =");
        nensu = s.nextInt();
        System.out.println("Please Enter nenri (%) =");
        nenri = s.nextDouble();
        r = 1 + (nenri * 0.01);
        ganri = (int)Math.floor(gankin * Math.pow(r, nensu));
        System.out.printf("ganri = %d (en)¥n", ganri);
        System.out.printf("risoku = %d (en)¥n", ganri - gankin);
    }
}
```

キーボードからの
データの読み込み

計算

画面表示

実行結果の例

```
Please Enter gankin (en) =  
10000  
Please Enter nensu (en) =  
10  
Please Enter nenri (%) =  
2  
ganri = 12189 (en)  
risoku = 2189 (en)
```

複利の計算



- 複利の公式：

$$\text{元利} = \text{元金} \times (1 + \text{年利})^{\text{年数}}$$

- **べき乗 x^y の計算のために, pow メソッドを使用する**

```
ganri = trunc( gankin * Power( r, nensu ) );
```

r は年利

floor



- floor

浮動小数点数の小数点以下を切り捨て

例) $3.4 \rightarrow 3$, $3.6 \rightarrow 4$, $-1.6 \rightarrow -2$

演習. 時間の換算



- **秒数 x** を読み込んで, **h 時, m 分, s 秒**を計算するプログラムを作りなさい.

例) $x=3723$ のとき,

$h = 1, m = 2, s = 3$