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

内容

- 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

目標

- 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

整数と浮動小数点数

- **整数**

  整数（正か負か０）

  例）

  ０

  ３

  ２８

  ４７７８

  －１

  －１０

  －１２５０

- **浮動小数点数**

  小数付きの数も**可**

  例）

  ０

  ３

  １２７８７４８６２３

  －４５６３７５９３９８

  ２．１９０８７２

  ０．０００１７８

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

オンライン開発環境 Online GDB

- **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

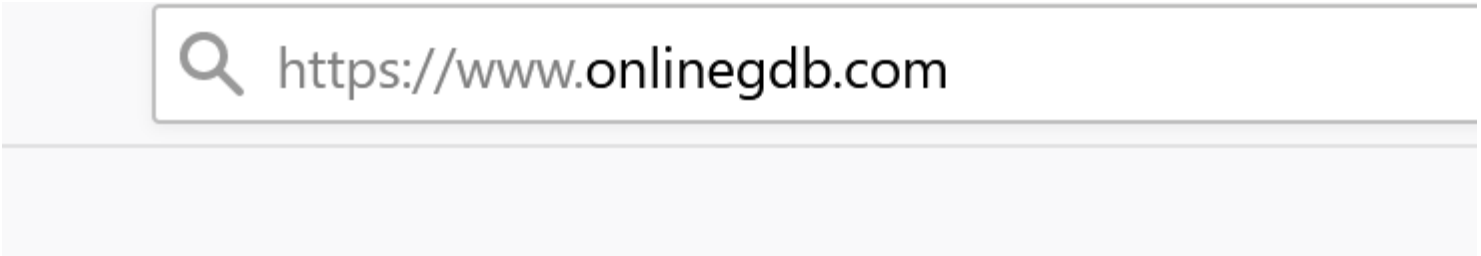
- オンラインなので、「秘密にしたいプログラム」を扱うには十分な注意が必要

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

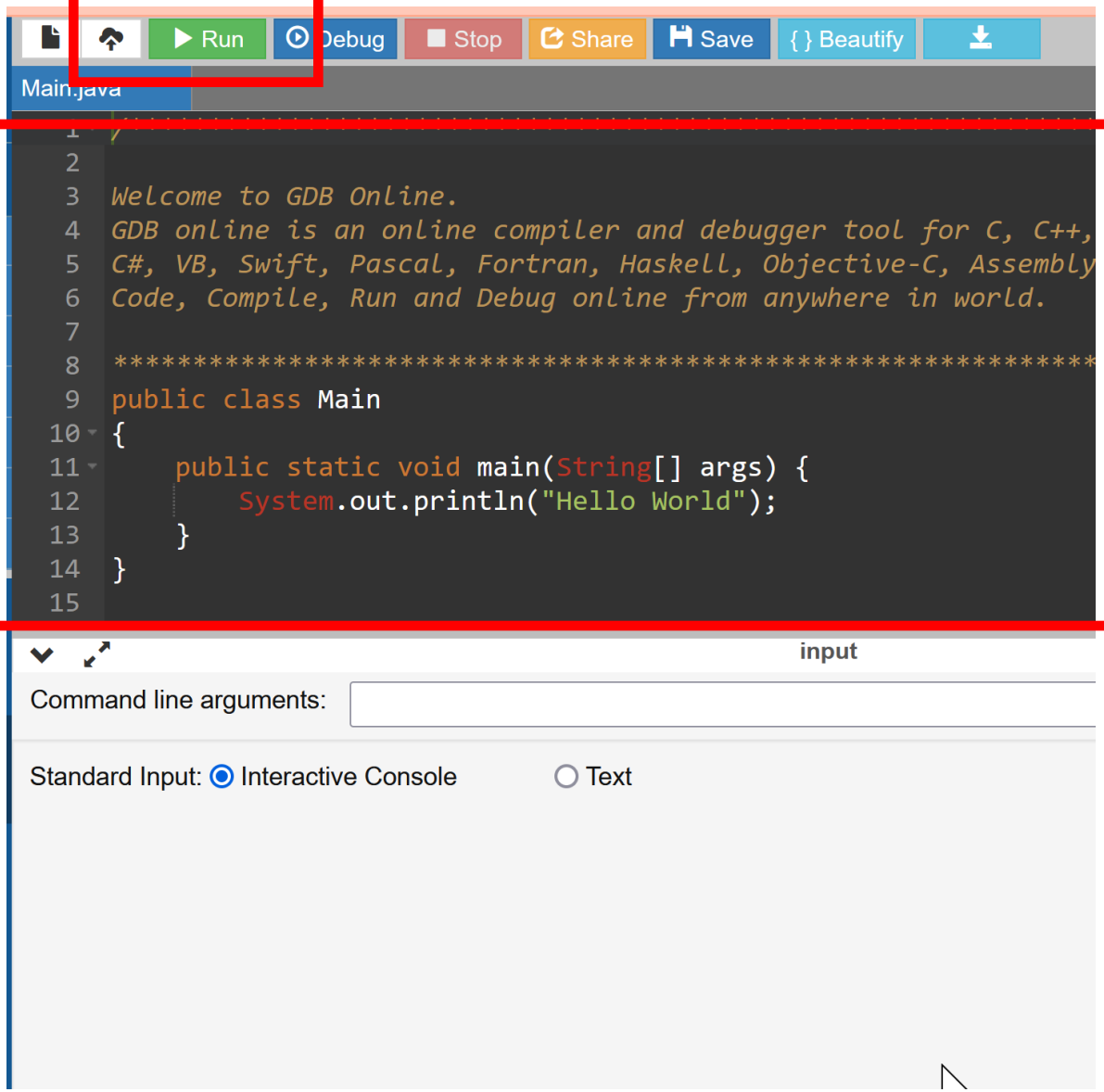


③「Language」のところで，「Java」を選ぶ

実行ボタン

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７６８円のとき，

５００円玉： １枚

１００円玉： ２枚

５０円玉： １枚

１０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

- １０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

硬貨の金種計算

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

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

メッセージを表示

```
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();
        if (((y % 400) == 0) || (((y % 100) != 0) && ((y % 4) == 0))) {
            System.out.printf("%d is leap year¥n", y);
        } else {
            System.out.printf("%d is not leap year¥n", y);
        }
    }
}
```

条件式

条件が成り立つ場合に実行される部分

条件が成り立たない場合に実行される部分

うるう年の判定

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) || (((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**

かつ

または

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（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

硬貨の金種計算

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：

$$元利 = 元金 \times （1 + 年利）^{年数}$$

- **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 → 3，3.6 → 4，　-1.6 → 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