2-1.データ型・文字列クラス

提出する 評定を受ける

■データ型・文字列クラス

はじめに

データ型とは、プログラムで使用する **箱の種類** の事を指します。

プログラムでは、値は **箱** に入れて使うことが多いです。
日常生活においても、 **小物を入れる箱、食材を入れる箱、衣服を入れる箱** など、色々な種類の箱がありますね。
そのようなイメージで、プログラムでも値を入れるために箱を使用しますし、入れる中身によって違う箱を準備します。

この章では、javaにはどんな **箱の種類(データ型)** があって、
そこにはどんな **値** を入れることができるのかを詳しく学んでいきましょう！

step1: データ型と値の関係性

javaで扱われる **データ型** の一覧表です。

種類	データ型名	扱える値・範囲
文字列クラス	String	"java"、"勉強"、"Mac" などの複数の文字を扱える
整数型	int	-2147483648 ～ 2147483647 までの整数を取り扱える
論理型	boolean	true (真) or false (偽) 真偽を測る
整数型	byte	-128 ～ 127 までの整数を取り扱える
整数型	short	-32768 ～ 32767 までの整数を取り扱える
整数型	long	-92223372036854775808 ～ 9223372036854775807 までの整数を取り扱える
浮動小数点型	float	± 3.4 × 1039 ～ ± 1.4 × 10 - 45 細かい数字を扱える
浮動小数点型	double	± 1.8 × 10308 ～ ± 3.4 × 10 - 324 さらに細かい数字を扱える
文字型	char	"あ" 、 "虹" などの一文字を扱える

解説

多くの種類がありますが、データ型の中で特に重要なのが、`String`と`int`です！
まずはこの2つさえ覚えれば問題ありません。
他のデータ型については、、このような種類があるんだ、となんとなく認識して頂ければとりあえずOKです。
それらのデータ型が出てきた時は、その時に改めて調べてみましょう。

Step2: データ型の使い方

Step1では、 **データ型(箱の種類)** と **値** の説明でしたが、
今度は使い方を学びましょう！

変数

?

データ型と値を扱うためには、
変数 というものについて触れる必要があるので順を追ってみていきましょう！

以下、プログラミング言語における「変数」の特徴です！

- 変数は、情報の出し入れを行える **入れ物（例: 箱のようなもの）**
- 変数の中身は、 **書き換え（or 上書き）が可能**
- 変数は、 **情報を一定期間記憶しておける**
- 変数には、 **名前を付与** する

上記を踏まえた上で、データ型や値と関連性を持たせ噛み砕いていくと、、、

・「何かの数値や、文字などの値にキーワードを付与して、 一時的に保存しておける」考え方

と紐付けることができます。

例題

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

        //int型を宣言して、「seisu」という変数名をつけて、1234で初期化する。

        int seisu = 1234;

        System.out.println("int型 = 整数 = " + seisu);

    }

}
```

【実行結果】

```
int型 = 整数 = 1234
```

解説

上記の例題にある、`int seisu = 1234;` を分解して解説していきます。

・変数の宣言
これから何のデータ型を作るのかの宣言になります。
今回は`int`型で宣言しているため、「int型」の「seisu」という変数を扱うよ！
初期値は「12345」だよ！ということになります。

・変数名
`seisu`はデータ型の名前で、データ型には必ず名前をつける必要があります。
どういう名前をつけるかは自由です。
自由ですが、何を入れてるものなのか分かりやすい名前をつけましょう！

【サンプル: 命名規則 その１】

```
// 良い例
String fruit = "りんご";

// 悪い例
String vegetables = "牛肉";
```

悪い例は、明らかに 箱の名前(変数名) と 値 が噛み合っていません。
変数の宣言では、こういった **矛盾や意味の齟齬が発生しないような命名が大切** になります。

【サンプル: 命名規則 その２】

```
// 良い例
int lineNumber = 123;

// 悪い例
String linenumber = "456";
```

こちらは何が悪いかというと、
変数名で使用されている英単語がすべて小文字であるため、変数名の意味が一瞬分かりづらくなっている点です。

基本は、単語ひとつひとつの意味がわかりやすいように、
2つ目以降の単語は、頭文字を大文字にして記述 していきます。
（※ラクダのコブのような見た目から、 **キャメルケース** と呼ばれる書き方になります。

変数の値の上書き

既に定義して値が入っている変数に別の値を格納したい場合は、
intやStringなどのデータ型は書かずに
「 `定義した変数 = 格納したい値` 」という書き方で実現できます。

例題

```
public class Curriculum {

                        public static void main(String[] args) {

        // Stringクラスを宣言して、「result」という変数名をつけて、「アタリ」で初期化する。
        String result = "アタリ";
        System.out.println(result);


        // 変数「result」に「ハズレ」という値を代入して上書きする。
        result = "ハズレ";
        System.out.println(result);
                }
}
```

【実行結果】

```
アタリ
ハズレ
```

・`;`（セミコロン）
これは句点(。)の意味があります。
忘れずに、最後は `;` で閉めましょう。
セミコロンが無い場合は、コンパイルエラーが発生します。
（※コンパイルエラーは、 **後述のStep4: エラー（コンパイルエラー）** で解説します。

`データ型 + 変数名 + = + 値` という順番で成り立ってることが分かります。
基本的な構造なので、しっかり抑えておきましょう。

System.out.println()について

例題にて `System.out.println("int型 = 整数 = " + seisu);` という記述があります。
printlnメソッド などと呼んだりしますが、何を表示させるか指示できるものです。英語の意味そのままですね。

`()` の中に表示してもらいたいものをいれます。

注目していただきたい部分は `()` の中に `seisu` という記述があることです。
直訳すると、 **「seisuを表示する」** となります。
ここで言う **seisu** とは何か…

```
int seisu = 1234; //seisuという変数に1234が入っている
```

だから、【実行結果】の時に、`int型 = 整数 = 1234`となります。

`System.out.println()` については、Javaのコードの中で何度も出てきます。

補足

・`System.out.print()`
類似機能として、 `System.out.print()` が存在します。
`System.out.println()` との違いは、コンソールへの出力時に、出力内容の末尾に **改行文字が入るか入らないか** という点です！
（メソッド名の `ln（エルエヌ）` が **line** という単語を指し示していて、使用者はそこで判別して使い分けることができます。

・改行文字
Windows: `¥n（円マーク + n）`
Mac: `\n（バックスラッシュ + n）`
文字列中に上記改行文字を入れるなどして、 `System.out.print()` などで出力すると改行されていることを確認できます。

【サンプル】

```
System.out.print("ABC\nDEF")
```

【出力結果】

```
ABC
DEF
```

Step3: 文字列・Stringクラス ー【String】

javaには複数の文字を代入できる **文字列クラス** の `String` というものがあります。
（※ `String` に関しては、 他のデータ型と違い、 **データ型ではなくクラス** になります。

データ型の `char` でも文字を扱えると学びましたが、一文字しか扱えません。
なので複数の文字を扱える`String` の方が圧倒的に使いやすいです。

文字(文字列) を扱う時は、`String` クラスを使ってきましょう！

例題

```
public class Curriculum2 {
    public static void main(String[] args) {
        // Stringクラスを宣言して、変数名はmojiで「javaはプログラム言語」で初期化
        String moji = "javaはプログラム言語";
        System.out.print(moji);
    }
}
```

【実行結果】

```
javaはプログラム言語
```

解説

お気づきの方もいるかと思いますが、使い方はデータ型の使い方と全く同じです。
文字列クラス も`Stringクラス` + `変数名` + `=` + `値` で成り立っています。

一つだけ注意していただきたいのが、`"javaはプログラム言語"` の `" "`の部分です。
文字をコードに記述する時は、 `" "(ダブルクォーテーション)` の間に文字を記述するのがお決まりです。
忘れずに覚えておきましょう！

Step4: エラー（コンパイルエラー）

下記の場合はエラーとなります！型や値には注意しましょう。

エラーになる記述例

```
// 一文字じゃないのでエラー
char c = 'ad';
```

```
//「true」か「false」しか入らないのでエラー
boolean b = 1234;
```

```
// 扱える範囲を超えているのでエラー
byte by = -500;
```

```
// 少数点を扱えないので、エラー
int i = 3.442;
```

上記はいずれも「データ型」と「値」の関係に対するエラーです。
エラーになる場合は、何かしらの **間違い** が必ずあります。
データ型・クラスに対して、扱える値は決まっているので注意して記述してきましょう。

提出課題

提出ファイル

・Task2_1.java
2-1/Task2-1.java に課題がありますので、指示に従って記述して下さい。

評定概要

学生から秘匿	No
参加者	80
提出	55
要評定	0