Javaプログラミング実習

04. Javaプログラムの基本

株式会社ジードライブ

今回学ぶこと

- Javaプログラムを構成する基本要素とそれらに関する基本的な文法

Javaプログラムの例

- 単純なJavaプログラムの例

```
package com.example.hellojava;

public class Hello {

    public static void main(String[] args) {
        System.out.println("Hello, Java!");
    }

}
```

前章で作成したソースコードを元に、
プログラムの構造について見ていきます

ステートメント

- プログラムとして記述された命令文をステートメント(Statement)と呼ぶ
  - statementは、発言・声明・陳述などの意味

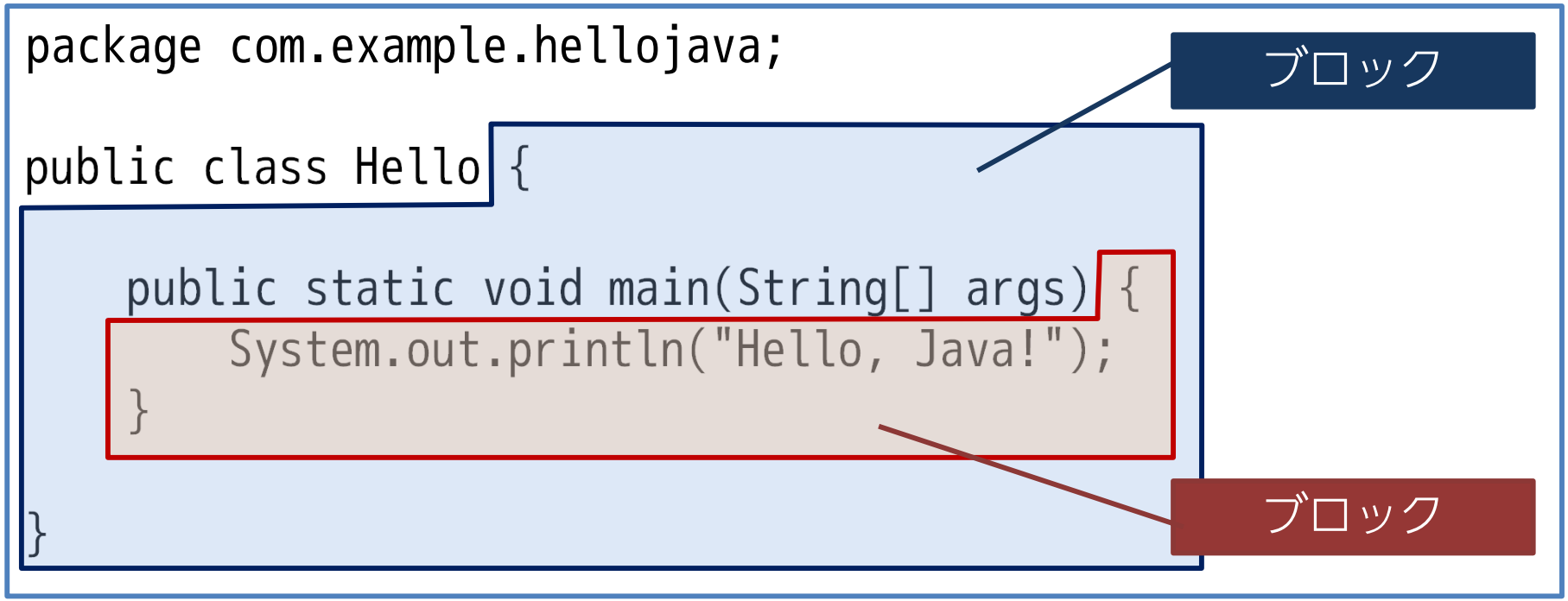
```
package com.example.hellojava;

public class Hello {

    public static void main(String[] args) {
        System.out.println("Hello, Java!");
    }

}
```

ステートメントに関するルール

- ステートメントの最後にはセミコロン(;)を付ける
- 複数のステートメントが並ぶ場合、基本的には、上から順番に処理される
- 計算や表示など、実際に何かを行わせる命令文のみがステートメントと呼ばれる
  - class の行や main の行はステートメントではない

ブロック

- 波カッコ { } で囲まれた部分をブロックと呼ぶ
  - ブロックはプログラム内の特定の区域を表す
  - ブロックの表す意味は色々ある(定義、分岐、反復等)



mainメソッド

- main( )という記述のブロックをmainメソッドと呼ぶ
- プログラム中には複数のメソッドを記述することができるが、実行時にはmainメソッドが最初に処理される

例

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, Java!");
    }
}
```

クラス

- Javaのプログラムはクラス(プログラムの部品)の集まりとして記述される
  - 複数のクラス (部品)を連携させて、１つのプログラムを構築する
    (小さなプログラムはクラスが1つということもあり得る)
  - クラス名とファイル名は一致させなくてはいけない

例：ファイル名が **Hello.java** の場合

```
package com.example.hellojava;

public class Hello {

    public static void main(String[] args) {
        System.out.println("Hello, Java!");
    }

}
```

クラス名

文字

- プログラム内でシングルクオート(')で囲まれた部分を文字という

```
package com.example.hellojava;

public class Hello {
    public static void main(String[] args) {
        System.out.print('H');
        System.out.print('e');
        System.out.print('l');
        System.out.print('l');
        System.out.println('o');
    }
}
```

文字列

- **"Hello, Java"** などプログラム内でダブルクオート(")記号で囲まれた部分を文字列という
  - プログラム内でテキスト情報を表す際に使用する
  - 正式には「文字列リテラル」という
  - 0文字でも、1文字であってもダブルクオートで囲めば、文字列データとして扱われる
    ⇒ 0文字の文字列("")を空文字という
- 文字列の規則
  - ダブルクオートで囲む
  - 文字列の長さは任意
  - 改行をそのまま含めることはできない

文字と文字列

- プログラム内でシングルクォート(')で囲まれた記述を文字という
- 文字と文字列は異なるデータとして扱われる
  - 例えば、**"A"** と **'A'** は異なるデータ
  - **'Hello'**のように、複数の文字をシングルクォートで囲むと構文エラーとなる

文字や文字列の出力

- 改行付きの文字列の出力
  - lnはline(行)という意味

```
System.out.println("Hello,");
System.out.println("Java!");
```

- 改行無しの文字列の出力

```
System.out.print("Hello,");
System.out.print("Java!");
```

System.out.println() は「sysout」と入力し、Ctrl + Space を押下することで簡単に入力できる

練習

- 練習04-1

エスケープシーケンス

- 特殊な文字を文字列の中に含める際に使用する表記法
- バックスラッシュ（ \ ）と文字を組み合わせる
  - 日本語環境ではバックスラッシュが円マーク(**¥**)で表示される
- 例えば、ダブルクオートを文字列の中で表現する場合は **¥"** と表記する

エスケープシーケンス

- エスケープシーケンスの例

エスケープシーケンス	対応する文字
¥'	'
¥"	"
¥¥	¥
¥n	改行
¥t	タブ

コメント

- プログラム内に記述するメモ書きで、プログラムの実行には影響しない
  - 複数行のコメントを記述するためのブロックコメントと１行のコメントを記述するための行コメントがある

```
package com.example.hellojava;
/*
  Helloプログラム
*/
public class Hello {
  public static void main(String[]
    // メッセージを表示
    System.out.println(“Hello, Java!”); // メッセージを表示
  }
}
```

ブロックコメント

- コメント開始記号(/*)から終了記号(*/)までの間がコメントとして扱われる
  - 複数行のコメントを作成することが可能
  - １行のコメントに利用しても構わない

```
/*
    Helloプログラム
    画面に「Hello, Java!」と表示させる
*/
public class Hello {
    public static void main(String[] args) {
    ... /* 1行分のコメント */
```

行コメント

- コメント開始記号(//)から行末までがコメントとして扱われる
  - 複数行のコメントを作成する場合は毎行頭にコメント開始記号を入れる

```
//  Helloプログラム
//  画面に「Hello, Java!」と表示させる
public class Hello {
    public static void main(String[] args) {
    ...
```

ブロックコメントと行コメントの使い分けに関する規則は特に無い

コメントアウト

- コードを削除せずとも、コメント化することで、無効にすることができる

```
/*
System.out.println("こんにちは。");
System.out.println("元気ですか？");
*/

System.out.println("Hello,");
// System.out.println("World!");
System.out.println("Java!")
```

実行されない

```
Hello,
Java!
```

行コメントの作成・除去のショートカットは Ctrl + /

フォーマット

- プログラムコードは適切な改行/空白/インデントを入れて見やすく記述することが望ましい
  - 見づらいコードはバグの元になる

```
package com.example.hellojava;

public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, Java!");
    }

}
```

インデント（字下げ：tabキー、または半角スペースで入れる）

Eclipseの自動フォーマット機能：
ソースメニュー ⇒フォーマット(Ctrl + Shift + F)

練習

- 練習04-2