上級プログラミング1（第3回）

工学部　情報工学科
木村昌臣

今日のテーマ

- Javaプログラミング入門(1)のつづき
  - 変数
  - 制御文
  - メソッド　　など
- 文字列の扱い

Java プログラミング入門(1) のつづき

[復習]プログラムの構成(1)

プログラムはクラスとして構成される

```
class SampleClass01{
        static int   i=10;
        public static void main(String[] args){
                add10();  System.out.println("i="+i);
        }
        public static void add10(){
                i+=10;
        }
}
```

[復習]プログラムの構成(2)

クラス≒構造体メンバ(フィールド)
＋メンバを操作する関数(メソッド)

```
class SampleClass01{
    static int  i=10;
    public static void main
        add10();  System.out.println("i="+i);
    }
    public static void add10(){
        i+=10;
    }
}
```

フィールド

メソッド

[復習]プログラムの構成 (アプリケーションの場合)

クラスのmainメソッドが実行される
(C言語のmain関数のように)

```
class SampleClass01{
        static int   i=10;
        public static void main(String[] args){
            add10(); System.out.println("i="+i);
        }
        public static void add10(){
                i+=10;
        }
}
```

実行例

```
class SampleClass01{
        static int i=10;
        public static void main(String[] args){
                add10();   System.out.println("i="+i);    }
        public static void add10(){
                i+=10;
        }
}
```

```
C:¥>  javac  SampleClass01.java
C:¥>  java SampleClass01
        i=20
C:¥>
```

ファイル
SampleClass01.class
が生成される

基本データ型

- クラス・オブジェクト以外のデータ型であり、C言語の型と似ている
  - 整数型: byte, short, int, long
  - 実数型: float, double
  - 文字型: char　　など
- ただし、文字列はStringクラスのオブジェクトとして扱う
  - C言語とは異なり、char型の配列は使わない（ずっと高機能）
  - これはあとで説明する

変数の宣言

- Javaでは、使用される前であればどこでも宣言可能

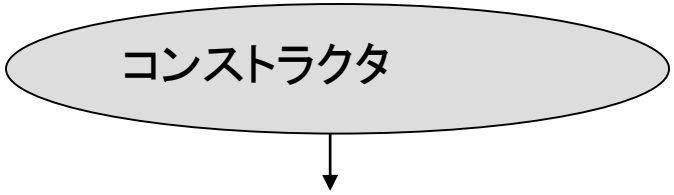
```
for(int i=0; i<10; i++){
    int j=2*i;
    a[i]=j;
}
```

演算子

- C言語と同じものが使える
  - 算術演算子 (＋ - ＊ /)
  - 代入演算子 (=)
  - 比較演算子 (＜ ＞ == !=)
  - 論理演算子 (&& ||)
  - インクリメント/デクリメント演算子 (++ --)　など
- Java特有の演算子もある
  - new演算子（オブジェクトや配列の生成）
  - アクセス演算子 (.)

コンストラクタ

- オブジェクトの初期化を行うための処理
- クラス名と同じ名前を持つ
- new演算子と組み合わせてオブジェクトを初期化
- デフォルトは引数なし
  - クラスにコンストラクタを定義しなければ引数なしのコンストラクタがデフォルトで定義される




```
Person kimura = new Person();
```

※定義の仕方は今後の回で説明する

アクセス演算子

- クラスやオブジェクトのメンバ(フィールドとメソッド)を使うときに使用

クラスが持つメソッドの呼び出し

Person.setDefaultLang(“Japanese”)

オブジェクトのメソッドの呼び出し

kimura.addLang(“English”)

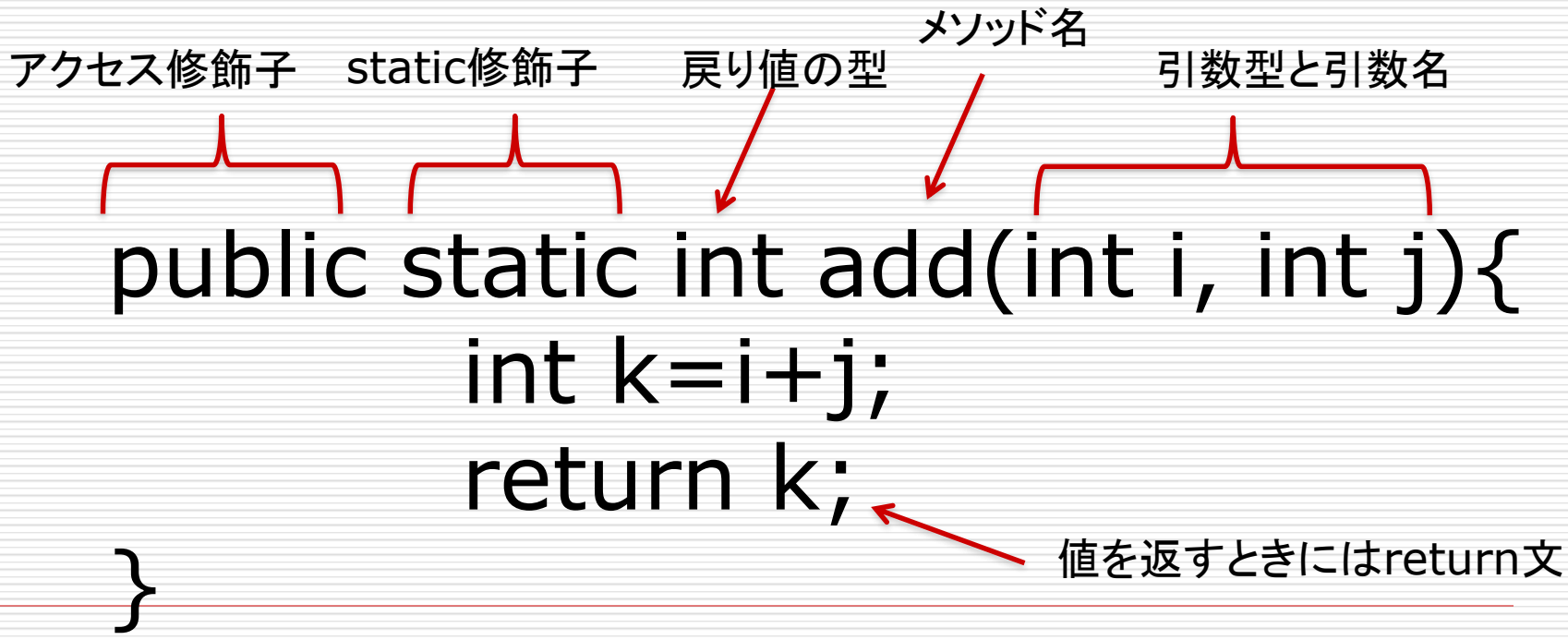
制御文

- 基本的にC言語と書式は同じ
  - if文 / switch文
  - for文 / while文
- ただし、必要な変数を、制御文の内部で宣言できる

```
for(int i=0; i<10; i++){
    a[i]++;
}
```

関数＝メソッド

- C言語の関数とほぼ同じ
- ただし、クラス内に記述する



アクセス修飾子

- 同一クラス（オブジェクト）外部で利用可能かを示すキーワード
  - public：どのクラス（オブジェクト）からも利用可能
  - private： 自クラス （オブジェクト）からのみ利用可
  - なし： 同じディレクトリ内のクラスからは利用可能
    - 正確には、同パッケージのクラスから利用可能
  - この他にprotectedがあり
- 詳しくは後の回で

static修飾子

- クラスから直接呼び出すメソッドを定義するときに、メソッド名の前に記述
  - オブジェクトで共通に使われるフィールド、メソッドにも

```
class Person{
    static defaultLang = “”;
    static setDefaultLang(String lang){
        defaultLang = lang;
    }
    …
}
```

配列

- 宣言の仕方はC言語と少し違う場合がある
  - あらかじめ要素を指定する場合は同じ
    int num[5] = {5, 3, 12, 0, 0};
  - あらかじめ領域を確保する場合は違うので注意
    int num[] = new int[5];
    もしくは
    int[] num = new int[5];
  - 要素数は
    num.length　←メソッドではない。
    かっこが付かないので注意

ArrayList

- 実際のJavaのプログラムでは配列はあまり使わず、代わりにArrayListクラスのオブジェクトを使う
  - ArrayList al=new ArrayList();
  - 次のようにするとA、B、Cの順番に要素が格納される
    - al.add(“A”);
    - al.add(“B”);
    - al.add(“C”);
  - java.utilパッケージをimportする必要あり
    - パッケージのimportについては今後説明します

文字列

- JavaではStringクラスのオブジェクト
  - char型の配列ではない
  - 高機能
    - 文字列の一部を別の文字列に置き換えることができる
    - 文字列の長さの情報を直接参照できる
    - ある文字を区切り文字にして文字列を分割することができる

などの機能を持っている

Stringオブジェクトの初期化

- 文字列で初期化
  - String s=“abcde”;
    - ダブルクォーテーションで囲まれた文字列はStringクラスのオブジェクトそのもの
- コンストラクタ(*)を使って初期化
  - char[] data={‘a’, ‘b’, ‘c’, ‘d’, ‘e’}
  - String s=new String(data);
    - シングルクォーテーションで囲まれたものは、char型

＊初期化処理をするもの。詳しくは次回で。

Stringクラスの主要なメソッド(1)

- String str=“abc-pbd”;
  - str.length()
    - 文字の長さを返され、この例の場合、7が返る
  - str.replace(‘b’,’e’)
    - 文字を置き換えた結果の文字列を返され、この例の場合、”aec-ped”が返る
  - str.split(“-”)
    - 引数の文字列を区切り文字として分割された文字列を配列として返され、この例の場合は{“abc”, “pbd”}が返る

Stringクラスの主要なメソッド(2)

- String str="abc-pbd";
  - str.indexOf("bc")
    - 文字列bdがstrに出てくる位置を返す。この場合、1が返される(先頭が0であることに注意)
    - 一度も出てこなければ-1を返す
  - str.compareTo("efg")
    - 辞書順で引数の文字列とくらべてstrの方が早ければ正の値、等しければ0、遅ければ負の値を返す
  - str.equals("abc-pbd")
    - 同じ文字列ならtrue、そうでなければfalse

文字列の演算

- 文字列は足し算ができる
  String str1=“abc”;
  String str2=“def”;
  とすると
  String str=str1 + str2;
  は、“abcdef” となる
  当然ながら “abc” + “def” という演算もできる
  （結果は同じ）
- str1+3.14 とすると “abc3.14”という結果に

文字列を繰り返し足してみる

- 例）

```
String str="abc";
for(int i=0; i<10000; i++){
    str = str + "abc";
}
```

確かに足し算の処理は出来るが、もともとStringクラスは固定文字列を扱うためのものであるため、このような処理についてはパフォーマンスが悪い（処理時間が長くかかる）

可変文字列を扱うには

- StringBufferクラスを使う
- 例）

```
StringBuffer strBuff
                =new StringBuffer(“abc”);
for(int i=0; i<10000; i++){
    strBuff.append(“abc”);
}
String str=strBuff.toString();
```

実は

- Stringオブジェクトの足し算では、処理時に暗黙的にStringBufferへの変換が行われている
- str = str + “abc”;

は以下の処理と同等（一回の足し算ごとに毎回StringBufferオブジェクトが生成されてしまう*）

```
StringBuffer sb = new StringBuffer(str)
sb.append(“abc”)
str=sb.toString();     /*Stringオブジェクトへの変換*/
```

*オブジェクトの生成と解放は比較的重い処理であることに注意

mainメソッドの引数について

```
class SampleClass01a{

        public static void main(String[] args){
                print(args[0]);
        }

        public static void print(String str){
                System.out.println(str);
        }
}
```

ここにStringクラスの配列を引数にとる宣言がある

C言語のargv, argcの代わりが、Javaのargsとなる

mainメソッドの引数

- 先ほどのクラスをコンパイルしておく

  javac SampleClass01a.java

- その後、下記のように実行したとする

  java SampleClass01a 芝浦工大 工学部

  args[0] args[1]

  クラス名の後ろにある文字列はmainメソッドに対して引数として渡される

- 前ページのプログラムでは「芝浦工大」が画面に表示される