静的インポート宣言

クラスの『型』だけではなく、クラスの『静的なメンバ』である、以下の二つのものもインポートできるようになっています。

- クラス変数（静的フィールド）
- クラスメソッド（静的メソッド）

これらのインポートを行うのが、**静的インポート**（*static import*）と呼ばれるインポートです。

型インポートの宣言と同様に、静的インポートの宣言にも２種類があります。宣言の形式は、次のようになります。**static** が付くことに注意しましょう。

```
import static パッケージ名 . 型名 . 識別子名 ;   単一静的インポート宣言
import static パッケージ名 . 型名 . *;          オンデマンド静的インポート宣言
```

静的インポートを利用したプログラム例を **List 11-3** に示します。

List 11-3 Chap11/Circle2.java

```
// 円の面積を求める（円周率Math.PIを静的インポート）

import java.util.Scanner;
import static java.lang.Math.PI;

class Circle2 {

   public static void main(String[] args) {
      Scanner stdIn = new Scanner(System.in);

      System.out.println("円の面積を求めます。");
      System.out.print("半径：");
      double r = stdIn.nextDouble();
      System.out.println("面積は" + (PI * r * r) + "です。");
   }
}
```

実行例

```
円の面積を求めます。
半径：5.5↵
面積は95.Ø3317777ØØ9123です。
```

本プログラムでは、**java.lang.Math** クラスに所属するクラス変数 PI（p.342）、すなわち **java.lang.Math.PI** を静的インポートした上で、単純名 PI でアクセスしています。

＊

前章で学習したように、**Math** クラスは、三角関数を計算する sin，cos，tan や、絶対値を求める abs など、数多くのクラスメソッドを提供します。

その中の三つのメソッドを呼び出すプログラム例を、**Fig.11-4** a に示します。

すべてのメソッド呼出しに **Math.** が付いています。このように、**Math** クラスに所属するメソッドを何度も呼び出すプログラムは、図 b のように実現するとよいでしょう。オンデマンド静的インポート宣言を使えば、個々のメソッドを別々にインポートすることなく、単純名であるメソッド名だけで呼び出せるようになります。

a 静的インポート宣言なし

```
// 静的インポートしなければ…
// ...
x = Math.sqrt(Math.abs(y));
z = Math.sin(a) + Math.cos(b);
```

すべてのメソッド呼出しに Math. が必要。

b 静的インポート宣言あり

```
import static java.lang.Math.*;
// ...
x = sqrt(abs(y));
z = sin(a) + cos(b);
```

Fig.11-4　クラスメソッドの静的インポート

重 要　特定のクラスに所属するクラス変数またはクラスメソッドを多用するプログラムでは、オンデマンド静的インポート宣言を行うとよい。

画面への表示とキーボードからの読込みで利用する **System.out** と **System.in** は、**System** クラスに所属するクラス変数です。それらを静的インポートすると、単純名 out と in とでアクセスできます。**List 11-4** に示すのが、そのように実現したプログラム例です。

▶　本プログラムは、文法を理解するためのものです。ただの out と in では、何のことだか分からないため、このようなプログラムを書くことはお勧めしません。

List 11-4　Chap11/Circle3.java

```
// 円の面積を求める（System.inとSystem.outを静的インポート）

import java.util.Scanner;
import static java.lang.Math.PI;
import static java.lang.System.in;
import static java.lang.System.out;

class Circle3 {

   public static void main(String[] args) {
      Scanner stdIn = new Scanner(in);
      out.println("円の面積を求めます。");
      out.print("半径：");
      double r = stdIn.nextDouble();
      out.println("面積は" + (PI * r * r) + "です。");
   }
}
```

実行例

```
円の面積を求めます。
半径：5.5
面積は95.03317777109123です。
```

なお、プログラムを以下のように実現することはできません。

```
import static java.lang.System.out.println;      // コンパイルエラー
//...
println("円の面積を求めます。");
```

このプログラムがエラーとなる理由は単純です。println がクラス（静的）メソッドではなく、インスタンスメソッドだからです。

▶　**System.out.println** の個々の要素は、以下のようになっています。

System	… **java.lang** パッケージに所属するクラス。
System.out	… **System** クラスのクラス（静的）変数（型は ***PrintStream*** クラス型）。
System.out.println	… ***PrintStream*** クラスのインスタンスメソッド。