

## Ph265. Java data types, flow control, packages, imaging

### Data types

Java is much closer to computer native data than Maple is. While Maple operates in decimal numbers, Java native data uses binary system. One binary digit (or bit) is either 0 or 1. A group of 8 bits is called byte.

There are several types of integers one can use in Java. The main difference is the maximum number one can store, and amount of memory used. All integers use the same format for storing the numbers. Negative numbers typically stored in “complementary” format.

- Integer data types include: byte (1 byte, max value is 128), short (2 bytes, max value 326768), int (4 bytes, max value around  $2 \cdot 10^9$ ), and long (8 bytes, max value around  $10^{18}$ ).
- Boolean data may have only two values – it is either true or false.
- Character takes two bytes – all characters are Unicode.
- Two kinds of floating points provided: the 32-bit (4 bytes) float, and 64 bit (8 bytes) double.

Two typical errors with floating point include overflow and underflow (when the exponent of the number) goes beyond its allowed values. Another type of errors typically associated with floating point numbers is round-off error – which typically lead to a loss of precision.

```
1  /* Limits.java: Determine machine precision */
2  public class Limits
3  {   public static void main(String [] args)
4  {   final int N = 60;           //Declaration
5      int i;                     //Declaration
6      double eps = 1.0, one_Plus_Eps; //Declaration
7      for (i = 0; i<N; i=i+1)
8      { eps = eps/2.;
9        one_Plus_Eps = 1.0 + eps;
10       System.out.println("one + eps = " + one_Plus_Eps + ", eps =
11         " +eps);   }
12     }}                          //End main, then class
13  /* OUTPUT:
14  one + eps = 1.5, eps = 0.5
15  one + eps = 1.0000000000000002, eps = 2.220446049250313E-16
16  one + eps = 1.0, eps = 1.1102230246251565E-16 */
```

### Basic operations in Java:

- The assignment operator: =
- The comparison operators: ==, >=, <=, >, <, !=
- Logical operators: ||, &&, !
- Arithmetic operators: +, -, \*, /, ++, --, +=, -=, \*=, /=;

## Functions in Java

A function in Java is defined by its name and signature – the number and types of its arguments. For example, consider the following code and guess which function is called in lines 11-13. Keep in mind that Java does not like to decrease the precision of calculations. Also note the local and global variables:

```
1 //Overload.java: method overloading
2 public class Overload {
3     //Main method does summation
4     public static void main(String[] argv) {
5         double x = 2., y = 4., z;
6         int i=3, j=5, k;
7         //Test choice by argument number
8         z = f(x);
9         z = f(x,y);
10        //Test choice by argument type/number
11        k = f(i);
12        k = f(i,j);
13        z = f(i,x);
14    }
15    public static double f(double a) {
16        System.out.println("1 double arg, return double");
17        return a*a; }
18    public static double f(double a, double b) {
19        System.out.println("2 double arg's, return double");
20        return a*a + b*b; }
21    public static int f(int a) {
22        System.out.println("1 int arg, return int");
23        return a*a*a; }
24    public static int f(int a, int b) {
25        System.out.println("2 int arg's, return int");
26        return a*b; }
27    public static int f(int a, double b) {
28        System.out.println("1 int arg + 1 double arg, return int");
29        return (int) a; }
30 }
```

## Flow control:

Is used in the programs to control the behavior of the calculations depending on the conditions realized in the particular experiment. Each of these structures have the format: Structure name (condition) statement; Note that Java gives you a way of combining separate statements into a single “compound statement” via curly brackets.

Name	Example	Comment
if-then	<code>if ( ( x &lt; 3 ) &amp;&amp; ( y == 12 ) ) z = y*x;</code>	One-way if.
if-then-else	<code>if ( x &lt;= 0.) {y = y * y;} else y = 2 * y;</code>	Two-way if. the catchall
if-then-else-if	<code>if (score &gt;= 90) {grade='A';} else if (score &gt;= 80) {grade='B';} else if (score &gt;= 70) {grade= 'C';}</code>	Multi-way if Any else-if's OK Inaccessible if other else true
for	<code>for (count=0; count&lt;100; count=count+1) {&lt;statements&gt;}</code>	Initial; for; increment Multi-line code block
do	<code>do {&lt;statements&gt;} while (&lt;boolean&gt;); &lt;labelname&gt;: x = x*y; ... break &lt;labelname&gt;; if (i==99) continue;</code>	Goes through at least once Control sent to here go to labelname Jump to loop end
do while	<code>while (Math.sin(x) &lt; 100.) {y = y*y; x = x + y;}</code>	Evaluate as long as true; line to execute
switch	<code>switch (month) {case 1: s="Jan"; break; case 12: s = "Dec"; break;}</code>	Case 1, break after code block Can have many cases

### Example:

```

1 //Switch.java: Demo use of switch control structure
2 public class Switch
3 {   public static void main(String[] argv)
4   {   int wake;                               //Declaration
5       for(int month = 12; month > -1; month--)
6       { System.out.println("In for loop month =" +month);
7         switch (month) {                       //switch structure
8           case 0:
9             System.out.println("last year");
10            break;                               //break
11           case 3:
12            wake = 7;
13            System.out.println("February, wake =" +wake );
14            break;
15           case 6:
16            wake = 6;
17            System.out.println("March, wake =" +wake );
18            break;
19           case 12:
20            wake = 9;
21            System.out.println("January, wake =" +wake );
22            break;
23           default: wake = 8;                     //default
24            break;                               }
25       }
26 }}                                             //End main, end class

```

## Packages

You can group classes you write into a package. Also, you can use previously written classes and packages in the new program. Before using these classes, we must “include”

their definition into our program. This is achieved via “import” operator. For example, consider the basic graphics package:

```
1 // EasyPtPlot.java: A simple plot application, plots f(x)
2 import tolemy.plot.*; //Plotting package
3 public class EasyPtPlot
4 { public static final double Xmin = -5., Xmax = 5.;
   //y range automatic
5 public static final int Npoint = 500;
6 public static void main(String[] args)
7 { Plot plotObj = new Plot(); // Create ptPlot object
8   plotObj.setTitle("f(x) vs x");
9   plotObj.setXLabel("x");
10  plotObj.setYLabel("f(x)");
11 // Format: addPoint(int dataSet, double x, double y, boolean
   connect)
12 double xStep = (Xmax - Xmin) / Npoint;
13 for (double x = Xmin; x <= Xmax; x += xStep)
14 { double y = Math.cos(x);
15   plotObj.addPoint(0, x, y, true); } //Add point
16 PlotApplication app = new PlotApplication(plotObj); //Display
17 }}
```