Reference Types

Data of reference type **must** be instantiated before it can be used.

A variable of **reference type** refers to (is bound to) an object

Examples:

- JFrame
- Color
- Rectangle, etc.

Primitive Types

Data of primitive type **can not** be instantiated.

A variable of primitive type has a value.

Primitive types generally permit a built-in Java notation for constants (sometimes called **literals**).

Primitive types often support operators.

Example primitive types:

```java
int
boolean
char
```
Java Primitive Types

Integer numeric types

- **byte** -- one byte
  - range: -128 through 127
- **short** -- two bytes
  - range: -32678 through 32677
- **int** -- four bytes
  - range: -2147483648 through 2147483647
- **long** -- eight bytes
  - range: -9223372036854775808 through 9223372036854775807

Real numeric types

- **float** -- four bytes
  - accuracy: 7 decimal digits
- **double** -- eight bytes
  - accuracy: 15 decimal digits

Logical type

- **boolean** -- one byte

Character type

- **char** -- two bytes

These are discussed later

Constant Syntax

- one or more consecutive decimal digits (0-9), optionally preceded by - or +
  - examples: 251, -17, 0, 7012903

Prefix Operator (abridged)

- unary negation
  - example: -7

Infix Operators (abridged)

- addition
  - example: -75 + 23
- subtraction
  - example: 127 - 16
- multiplication
  - example: 12 * 5
- integer division (i.e., quotient)
  - example: 12 / 5
- remainder
  - example: 11 % 3

Postfix Operators (abridged)

- autoncrement
  - example: someIntVar++
- autodecrement
  - example: someIntVar--
**Constant Syntax**
- A number with an embedded decimal point; leading - or + is optional; E-format can be used for scientific notation.
- Examples: 25.1, -.17, -34.2E17, 5.2E+3

**Prefix Operator** (abridged)
- Unary negation
  - Example: -2.5

**Infix Operators** (abridged)
- Addition
  - Example: -7.5 + 2.3
- Subtraction
  - Example: 18.7 - 16.
- Multiplication
  - Example: 10.1 * 0.5
- Real division
  - Example: 14.4 / 1.2

---

**Question**
How is the following expression evaluated?

7 + 3 * 2

Is it (7+3) * 2 OR 7 + (3*2)???

**Answer** (informally)
Java uses rules that are similar to those understood by mathematicians.

**Evaluation Rules**
1. Operations inside parentheses are performed before those outside.
2. Within the same parenthesis grouping operations of higher precedence are performed before those with lower precedence.

<table>
<thead>
<tr>
<th>Precedence Table (abridged)</th>
</tr>
</thead>
<tbody>
<tr>
<td>++ -- postfix</td>
</tr>
<tr>
<td>- unary</td>
</tr>
<tr>
<td>* / % multiplicative</td>
</tr>
<tr>
<td>+ - additive</td>
</tr>
</tbody>
</table>

3. Within equal precedence groupings operations are performed left to right (with a few exceptions, as shown in the complete precedence table).
Examples

\[
\begin{array}{c}
100 / 3 - 2 * 5 \% 3 + \neg(-6) * (1 + 1) \\
\quad 33 \\
\quad 10 \\
\quad 6 \\
\quad 2 \\
\quad 12 \\
\quad 1 \\
\quad 32 \\
\quad 44 \\
\end{array}
\]

\[
1 + 2 * 3 * 4 - 5 + 6 / 7 + 9 \% 2 \\
\]

\[
(1 + 2) * (3 + 1) * 2 - (6 * (7 + (8 * 2)))
\]

\[
(12.2 - 2.2) / (-3.3 + 5.3) + 0.4 * 10.0
\]

Primitive Variables

A variable of primitive type names a memory location that stores one value. The variable represents this value.

Example Declaration

```java
private int headCount;
private int footCount;
```

Example Instruction Sequence

```java
headCount = 7 + 3 * 2;
footCount = headCount * 2;
headCount = headCount + 1;
headCount++;
```
Mixing Numeric Types

It is acceptable to assign an expression of one type to a variable of wider type. Such assignments are considered to be safe.

Examples (assume each variable name indicates its type)

\[
\text{someDouble} = 25 \div 2.0; \\
\text{someInt} = \text{someByte}; \\
\text{someFloat} = 13 \div 3; \\
\text{someByte} = 35.2;
\]

Sometimes it is necessary to convert unsafely by narrowing

Example (The second instruction below results in a compile-time error.)

\[
\text{someDouble} = 25.0; \\
\text{someInt} = \text{someDouble};
\]

Casting Syntax

\[
(\text{TargetType}) \ \text{Expression}
\]

Semantics

A cast can be applied to many expressions to convert them to roughly equivalent values of a different type (called TargetType above).

Three acceptable casts

- cast for converting to a wider type \( \text{someDouble} = \)
- cast to convert double to float (some potential loss of accuracy) \( \text{someInt} = \)
- cast to convert float or double to int \( \text{someInt} = \)

\text{truncation means fractional portion eliminated}
Casting Precedence

Casts have a higher precedence than +, -, (subtraction), *, /, and %.

Example (The instruction below results in a compile-time error.)

\[ \text{someInt} = (\text{int}) 2 \times \text{someDouble}; \]

Solution: Use parentheses

\[ \text{someInt} = (\text{int})(2 \times \text{someDouble}); \]

\[ \text{OR} \]

\[ \text{someInt} = 2 \times (\text{int})\text{someDouble}; \]

Do these produce the same results?

Two Useful Java Features

Variable Initialization
Java permits a variable to be initialized by merging the declaration and assignment.

Examples

\[ \text{int depth} = ; \]

\[ \text{private Oval dot} = ; \]

The initialization must consist of a single assignment that can be executed at the time of declaration.

final Variables
A variable declared as final can be assigned a value just once.

Examples

\[ \text{final int inchesPerFt} = 12; \]

\[ \text{private final Oval smallDot} = \text{new Oval}(0, 0, 1, 1); \]

A final variable is a good way to create a program-declared constant.
Initializing a final variable when it is declared is wise.
Primitive Parameters

Java uses parameter passage by __________.

When the method is called...
1) Each actual argument expression is evaluated.
2) A copy of the value of each argument expression is assigned to its formal parameter.
   (Thereafter, the formal parameter behaves like a local variable.)

Example Method

```java
private void moveOvalTo( Oval a, int j) {
    j = j + 1;
    a.setLocation(j, j);
}
```

Sample Call

```java
circle = new Oval(10, 10, 100, 100);
someInt = 40;
moveOvalTo( circle, someInt );
```

Reference type parameters are passed the same way. The “value” of the parameter is an object binding (i.e. the address of the actual argument object).

Exercises

Write a method for each of the following.

- This method has an int parameter, p, and a double parameter, r, where p represents an initial amount of money (in cents) and r is an annual interest rate (r == 5.2 means 5.2% interest). Your method must return the value of this investment after four years of compounded interest.

Write method with the following signature.

```java
private void printScores( int q1, int q2, int q3, int e1, int e2, int f)
```

Where q1, q2 and q3 are raw quiz scores (out of 10 points), e1 and e3 are raw exam scores (out of 100 points) and f is the raw final exam score (out of 150 points).

This method should print the percentage for each score as well as a total percentage for the class based on 10% from quizzes, 60% from exams and 30% from the final.
More Exercises

Write a method for each of the following.

• Write method with the following signature.

  private void printAreas(Rectangle r1, Rectangle r2)

  Your method should print the area of each rectangle, along with the
  average area. (The average should be accurate to one decimal point.)

• Write method with the following signature.

  private double trianglePerimeter(int x1, int y1, int x2, int y2, int x3, int y3)

  where (x1, y1) and the coordinates of one vertex of a triangle, and the
  other two vertices are (x2, y2) and (x3, y3).

  This method should return the length of the perimeter for the triangle formed
  by these three vertices.