Chapter 3
Syntax, Errors, and Debugging


Lambert / Osborne
Objectives

- How do you construct and use numeric and string literals?
- How do you name and use variables and constants?
- What are arithmetic expressions?
- What is the order of precedence?
- How do you concatenate?
- How and when do you use comments in a program?
Objectives (continued)

- What are syntax errors, run-time errors, and logic errors?
- How do you insert output statements to debug a program?
- What is the difference between Cartesian coordinates and screen coordinates?
- How do you change color and text properties?
Vocabulary

- arithmetic expression
- comments
- coordinate system
- exception
- graphics context
- keywords
- literal
- logic error
- method signature
- origin
- package
Vocabulary (continued)

- pseudocode
- reserved words
- run-time error
- screen coordinate system
- semantics
- syntax
- syntax errors
- variable declaration statement
- virus
Language Elements

- Programming and natural languages share three elements.
  - **Vocabulary**: Words and symbols
    - **Syntax**: Rules for combining statements.
      - Use of operators, parentheses, etc.
    - **Semantics**: Rules for interpreting statements.
      - Order of precedence

<table>
<thead>
<tr>
<th>TYPE OF ELEMENT</th>
<th>EXAMPLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>arithmetic operators</td>
<td>+  -  *  /</td>
</tr>
<tr>
<td>assignment operator</td>
<td>=</td>
</tr>
<tr>
<td>numeric literals</td>
<td>5.73  9</td>
</tr>
<tr>
<td>programmer defined variable names</td>
<td>fahrenheit  celsius</td>
</tr>
</tbody>
</table>
Language Elements (continued)

- **Programming Vs. Natural Languages:**
- Programming and natural languages have three differences.
  - **Size:** small vocabulary, simple syntax.
  - **Rigidity:** Syntax must be absolutely correct.
  - **Literalness:** Computers follow exact instructions.
- It is difficult to express complex ideas using limited syntax and semantics of programming languages.
Basic Java Syntax and Semantics

• **Data Types:**
  - Primitive: numbers, characters, Booleans
  - Objects: scanners, strings, and more

• **Syntax:**

• Syntax differs for objects and primitive data.
  - Primitive: data types are combined in expressions with operators (addition, multiplication)
  - Objects: send messages, must be instantiated before use
Basic Java Syntax and Semantics (continued)

- **Numbers:**

- **Numeric data types:**
  - Most programs input numeric data, perform calculations, output numeric results.
  - Java includes six numeric data types but we use two:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>STORAGE REQUIREMENTS</th>
<th>RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>4 bytes</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes</td>
<td>-1.79769313486231570E+308 to 1.79769313486231570E+308</td>
</tr>
</tbody>
</table>
Literals:
Liters are items in a program whose values do not change.

- Restricted to primitive data types and strings.

<table>
<thead>
<tr>
<th>EXAMPLE</th>
<th>DATA TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>an integer</td>
</tr>
<tr>
<td>-31444843</td>
<td>a negative integer</td>
</tr>
<tr>
<td>3.14</td>
<td>a floating-point number (double)</td>
</tr>
<tr>
<td>5.301E5</td>
<td>a floating-point number equivalent to $5.301 \times 10^5$, or 530,100</td>
</tr>
<tr>
<td>5.301E-5</td>
<td>a floating-point number equivalent to $5.301 \times 10^{-5}$, or 0.00005301 (double)</td>
</tr>
</tbody>
</table>

Examples of numeric literals
Variables and Their Declarations:
- Variables are items whose values can change during execution.

Changing the value of a variable
Variables and Their Declarations (cont):

Before using a variable, the program must declare its type.

- Variable declaration statement
- Type on left; variable name on right

```java
int age;
double celsius;
String name;
Scanner reader;
```

- Several variables and values can be in the same statement.
Variables and Their Declarations (cont):

- Instantiation creates objects.

```
new <name of class> (<zero or more parameters>)
```

- Constants are variables that, once initialized, cannot change.

```
final double SALES_TAX_RATE = .0785;
```
Assignment statements have variables on the left and values on the right.

Arithmetic expressions are operands and operators.
  - Multiplication/division before addition/subtraction.
  - Equal operators calculated from left to right.
  - Use parentheses to change the order.
Explanation about points concerning operators:

- Division has different semantics for integer and floating-point operands.
- \% yields the remainder of one number divided by another.
- Java applies operators of higher precedence over lower precedence.
- Parentheses must occur in pairs, but can be nested.
Basic Java Syntax and Semantics (continued)

- Maximum, Minimum, and Arithmetic Overflow:
- Numeric data types support a finite range of values.
- Programmers use constants to represent value range.
  - Arithmetic overflow: when values are outside of range.
  - JVM inverts the sign of the number and allows the number to continue.
Basic Java Syntax and Semantics (continued)

- **Mixed-Mode Arithmetic:**
- In Java, integers are converted to doubles (floating-point) when mixed.

- **Casting to `int` and `double`:**
- Casting allows data types to be converted.
- The cast operator has higher precedence.
- Cast operators are useful for rounding floating-point numbers to integers.
Basic Java Syntax and Semantics (continued)

- **String Expressions and Methods:**
- Strings can be literals or assigned to variables.
- Strings can also be combined using concatenation operator and be sent messages.
  - Combine field names “first name” and “last name” to produce Bill Smith.
  - Strings can be concatenated to numbers.
Basic Java Syntax and Semantics (continued)

- **String Expressions and Methods (cont):**
- Escape character (\) is used to indicate that a quotation mark is to be taken literally, not as a delimiter.
  - Used to have commas and quotations in output.
  - Escape also used to indicate tabs (\t) and more.
  - If \ is needed in a string, use two (\\).
- A string returns its length in response to a length message.
Basic Java Syntax and Semantics (continued)

- **Methods, Messages, and Signatures:**
- An object can respond to a message only if its class implements a corresponding message (same name).
  - To use a method, you must know:
    - What type of value it returns
    - Its name
    - The number and type of parameters it expects
  - A method’s name and the types and numbers of its parameters are called the method’s signature.
User-Defined Symbols:

- Variable and program names are examples of user-defined symbols.
- User-defined symbols consist of a letter (A … Z), (a … z), (_ and $), followed by a sequence of letters and/or digits (0 … 9).

Names are case-sensitive.

Keywords and reserved words cannot be used as they have special meaning.
  - Else, byte, char, do, return, and more
Basic Java Syntax and Semantics (continued)

- Packages and the `import` Statement:
- Packages allow programmers to share code.
- Packages are collections of classes that can be imported into a program.
- An import statement form is: `import x.y.z;`
  - `x` is the package name
  - `y` is the subsection in the package
  - `z` is the class in the package
Objects support terminal I/O.
An instance of the Scanner class supports input and the object System.out supports output.
System.out is an instance of the class PrintStream.
  - This class and others are available to Java programmers without using import statements.
Terminal I/O for Different Data Types (continued)

- When a program encounters an input statement, it pauses and waits for the user to press Enter, then processes the user’s input.
- Interaction with user (bold) looks like this:

```
Enter your name (a string): Carole Jones
Enter your age (an integer): 45
Enter your weight (a double): 130.6
Greetings Carole Jones. You are 45 years old and you weigh 130.6 pounds.
```
Comments

- Comments are explanatory sentences inserted in a program used to clarify code and are ignored by the compiler.
  - End of line comments (followed by //</code>)
    
    ```java
    a = 3;       // assign 3 to variable a
    b = 4;       // assign 4 to variable b
    ```
  
  - Multi-line comments (opened by /** and closed by */)
    ```
    /* This code segment illustrates the
    use of assignment statements and comments */
    ```
To make a program readable:
- Begin with a statement of purpose.
- Use comments to explain variables’ purposes.
- Precede major segments of code with brief comments.
- Include comments to explain complex or tricky code sections.
Programming Errors

- The Three Types of Errors:

  - Syntax errors are detected at compile time.
    - Missing semi-colon or misspelling.

  - Run-time errors are when a computer is asked to do something illegal.
    - Divide by 0.
    - Null pointer: sending a message to uninstantiated object.
    - Java is case-sensitive, so Main and main are different.
The Three Types of Errors (cont):

- Logic errors occur when we fail to express ourselves accurately.
  - Tell someone to turn left, but we mean to say right.
  - Detected by incorrect output.
  - Use test data to compare output with expected results.
  - Desk checking: rereading code carefully.
Debugging

- After a bug is detected, you must find it.
  - Could reread the code, but the logic error is hard to find and detect.
  - Debug: Add lines of code to print values of variables in the terminal window, then run the program again.

Incorrect output from the temperature conversion program
Graphics and GUls: Drawing Shapes and Text

- **Defining a Specialized Panel Class:**
- An application window in a GUI program has a defined set of responsibilities.
- Before we create and display other objects, ask which object will be responsible for them.
  - The application window or the panel in which they appear
- Divide the labor and delegate responsibility.
  - Define a new panel by creating a new class
Graphics and GUls: Drawing Shapes and Text (continued)

- **Coordinate Systems:**
- Positions in this system are specified in terms of points with x and y coordinates relative to the origin (0,0).
- Java uses a screen coordinate system.
The Graphics Class:
The package `java.awt` provides a `Graphics` class for drawing in a panel.

A panel maintains an instance of this class, called a graphics context.
- Allows program to access and modify a panel’s bitmap

Each shape is drawn in a graphics context with a foreground color (black default).
### Sample method in the Graphics class:

<table>
<thead>
<tr>
<th>GRAPHICS METHOD</th>
<th>EXAMPLE CALL AND OUTPUT</th>
<th>WHAT IT DOES</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>drawArc()</code></td>
<td><code>g.drawArc(10, 25, 50, 50, 0, 90)</code></td>
<td>Draws an arc that fits within a rectangle whose upper-left corner is ((x, y)) and whose dimensions are the specified width and height. The arc is drawn from <code>startAngle</code> to <code>startAngle + arcAngle</code>. The angles are expressed in degrees. A start angle of 0 indicates the 3 o’clock position. A positive arc indicates a counterclockwise rotation, and a negative arc indicates a clockwise rotation from 3 o’clock.</td>
</tr>
</tbody>
</table>
Graphics and GUls: Drawing Shapes and Text (continued)

- **Drawing Shapes with the Method `paintComponent`:**
  - The responsibilities of a panel class:
    - Draw images in response to messages from the application
    - Redraw images when window is refreshed
  - When a window opens, the JVM sends the message `paintComponent` to each object.
    - Objects with images to draw do so
    - The application does not call `paintComponent`; it is triggered by the JVM in response to events.
Graphics and GUIs: Drawing Shapes and Text (continued)

- Finding the Width and Height of a Panel:
  - Find the width and height to center an image on a panel and keep it centered when window resizes.
  - Use `getWidth()` and `getHeight()`.

- Text Properties and the `Font` Class:
  - In a bitmapped display, text is drawn like an image.

<table>
<thead>
<tr>
<th>TEXT PROPERTY</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>Red, green, blue, white, black, etc.</td>
</tr>
<tr>
<td>Font style</td>
<td>Plain, bold, italic</td>
</tr>
<tr>
<td>Font size</td>
<td>10 point, 12 point, etc.</td>
</tr>
<tr>
<td>Font name</td>
<td>Courier, Times New Roman, etc.</td>
</tr>
</tbody>
</table>
In this chapter, you learned:

- Java programs use the `int` data type for whole numbers (integers) and `double` for floating-point numbers (numbers with decimals).

- Java variable and method names consist of a letter followed by additional letters or digits. Java keywords cannot be used as names.
Summary (continued)

- Final variables behave as constants; their values cannot change after they are declared.
- Arithmetic expressions are evaluated according to precedence. Some expressions yield different results for integer and floating-point operands.
- Strings may be concatenated to form a new string.
The compiler catches syntax errors. The JVM catches run-time errors. Logic errors, if they are caught, are detected by the programmer or user of the program at run time.

A useful way to find and remove logic errors is to insert debugging output statements to view the values of variables.
Summary (continued)

- Java uses a screen coordinate system to locate the positions of pixels in a window or panel. The origin of this system is in the upper-left corner or the drawing area, and the $x$ and $y$ axes increase to the right and downward, respectively.
- The programmer can modify the color with which images are drawn and the properties of text fonts for a given graphics object.