APPENDIX C
CODING STYLE

Although compilers ignore style, experience has shown that code that adheres to a simple and consistent style will actually contain fewer errors. Furthermore, such code is easier to read, and this makes modifying or debugging it much faster.

C.1 Naming Convention

1. **Constants**
   Use uppercase characters as in `LIMIT`. If the name is made up of more than one word, separate the words by an underscore as in `UPPER_LIMIT`.
2. **Variables, methods, and packages**
   Use lowercase characters as in `length`. If the name is made up of more than one word, capitalize the first letter of each subsequent word as in `indexOf`.
3. **Classes**
   Capitalize the first letter as in `String`. If the name is made up of more than one word, capitalize the first letter of each subsequent word as in `StringBuffer`.

The name of a constant or a variable must be indicative of content. Avoid using abbreviations (like `prc` for price) unless they are standard; and avoid using generic names (like `x` or `i`) unless the context itself is generic.

C.2 Statement Layout

1. **Editor**
   Use a non-proportional font like Courier with a point size of 10. Set the tab size to 3; i.e. a tab should skip two positions.
2. **One statement per line**
   Write only one statement per line. If a statement is longer than 80 characters then break it (ideally after a comma) and continue it indented on the next line.
3. **Between tokens**
   Leave exactly one space between tokens (e.g. around all infix operators like `*` and `=`) except in the following four situations (in which do not leave a space):
   - After an opening parenthesis
   - Before a closing parenthesis, comma, or semicolon
   - Around the dot separator
   - Between the name of a method and the opening parenthesis of its parameter list.
4. **Indentation**
   Indent each block by exactly two spaces relative to its braces.

C.3 Braces

1. **Usage**
   Use braces in control structures always, even if the block contains only one statement.
2. **Placement**
   Corresponding opening and closing braces must be vertically aligned.

C.4 Magic Numbers

Aside from zero, ±1, and ±2, no hard-coded literal (also known as *magic numbers*) may appear in code except in a `final` statement.

C.5 Examples

- Declaration

```
int counter;
final int PORT = 80;
final char CODE = '\u0041';
final char TAB = '\t';
double costPrice;
final double ELECTRON_CHARGE = 1.60219E-19;
```

- Expression

```
double z = Math.sqrt(x * x + y * y);
boolean isValid = (address > 0) && (address % size == 0);
```

- Selection

```
if (x % y == 0)
{  z = x + y;
} else if (x / 2 * 2 == x)
{  z = y;
} else
{  z = x;
}
```

- Iteration

```
int sum = 0;
for (int i = 0; i < name.length(); i++)
{  sum = sum + name.charAt(i);
}
```

- Exception Handling

```
try
{  int colon = entry.indexOf(":");
   IO.println(entry.substring(0, colon);
}
catch (IndexOutOfBoundsException e)
{  IO.println("No colon in input!");
}
```