Chapter 2: Java Fundamentals

Variables, Constants and Built-in Data Types
Objectives

• Discovering what is a variable
• Discovering what is a data type
• Learning about the basic data types
• Constants and variables identifiers
• Get acquainted with how to select proper types for numerical data.
• Write arithmetic expressions in Java.
Most programs require the temporary storage of data. The data to be processed is stored in a temporary storage in the computer's memory: space memory.

A space memory has three characteristics:
- Identifier
- Data Type
- State
State of the Space Memory

• The state of the space memory is the current value (data) stored in the space memory.

• The state of the space memory:
  • May be changed.
    – In this case the space memory is called variable.
  • Cannot be changed.
    – In this case the space memory is called constant.
Space Memory Identifier

- **Identifier** is a sequence of characters that denotes the name of the space memory to be used.
  - This name is unique within a program.

- **Identifier Rules**
  - It cannot begin with a digit (0 – 9).
  - It may contain the letters a to z, A to Z, the digits 0 to 9, and the underscore symbol, `_`.
  - No spaces or punctuation, except the underscore symbol, `_`, are allowed.
Identifier Conventions in Java

- Constants:
  - All uppercase, separating words within a multiword identifier with the underscore symbol, _.

- Variables
  - All lowercase.
  - Capitalizing the first letter of each word in a multiword identifier, except for the first word.
Identifiers are Case-Sensitive

• Identifiers in Java are case-sensitive. Thus, the identifiers myNumber and mynumber, are seen as two different identifiers by the compiler.
Data Type

- The **data type** defines what kinds of values a space memory is allowed to store.
- All values stored in the same space memory should be of the same data type.
- All constants and variables used in a Java program must be defined prior to their use in the program.
Java built-in Data Types

Constant or Variable

First Decision Level

Second Decision Level

Third Decision Level

Fourth Decision Level

Numeric

Integer

byte

short

int

long

Floating-point

float

double

Character

char

String

Boolean

boolean
# Primitive Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size (bits)</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>true, false</td>
<td>true, false</td>
<td>Stores a value that is either true or false.</td>
</tr>
<tr>
<td>char</td>
<td>16</td>
<td>-128 to +127</td>
<td>Stores a single 16-bit Unicode character.</td>
</tr>
<tr>
<td>byte</td>
<td>8</td>
<td>-128 to +127</td>
<td>Stores an integer.</td>
</tr>
<tr>
<td>short</td>
<td>16</td>
<td>-32768 to +32767</td>
<td>Stores an integer.</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648 to +2,147,483,647</td>
<td>Stores an integer.</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>accurate to 8 significant digits</td>
<td>Stores a single-precision floating point number.</td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>accurate to 16 significant digits</td>
<td>Stores a double-precision floating point number.</td>
</tr>
</tbody>
</table>
Variable/Constant Declaration

- When the declaration is made, memory space is allocated to store the values of the declared variable or constant.
- The declaration of a variable means allocating a space memory which state (value) may change.
- The declaration of a constant means allocating a space memory which state (value) cannot change.
Constant Declaration

```
final dataType constIdentifier = literal | expression;
```

Constants:
- `final double PI = 3.14159;`
- `final int MONTH_IN_YEAR = 12;`
- `final short FARADAY_CONSTANT = 23060;`
- `final int MAX = 1024;`
- `final int MIN = 128;`
- `final int AVG = (MAX + MIN) / 2;`

The reserved word `final` is used to declare constants.

These are constants, also called named constant.

These are called literals.

This is called expression.
Variable Declaration

• A variable may be declared:
  - With initial value.
  - Without initial value.

• Variable declaration with initial value;
  
  ```
  dataType variableIdentifier = literal | expression;
  ```

  ```
  double avg = 0.0;
  int i = 1;
  int x = 5, y = 7, z = (x+y)*3;
  ```

• Variable declaration without initial value;
  
  ```
  dataType variableIdentifier;
  ```

  ```
  double avg;
  int i;
  ```