



Introduction to Programming with Java™

OL302 VERSION 052802

Outsource Laboratories, Inc.
(Subsidiary of IGS, Inc.)

145 Wyckoff Road, 3rd Floor
Eatontown, New Jersey 07724

Toll Free: 888-GO-OLABS
Phone: 1 732 544-5001
Fax: 1 732 544-5002

URL: <http://www.olabs.com>
Email: training@olabs.com

For more information about Outsource Laboratories Professional Services, or to add your name to our monthly mailing list, please visit our website at <http://www.olabs.com>.

OUTSOURCE LABORATORIES

(Subsidiary of IGS, Inc.)

145 Wyckoff Road, 3rd Floor
Eatontown, NJ 07724

888-GO-OLABS toll free

732-544-5001 direct

training@olabs.com

732-544-5002 fax

<http://www.olabs.com>

This material is copyrighted by Outsource Laboratories © 2002. This material shall not be reproduced without the express written consent of Outsource Laboratories. All rights reserved.

Outsource Laboratories, Inc. and Olabs, Inc., are registered trademarks of Outsource Laboratories. All other products referenced herein are trademarks of their respective holders.

Copyright © 2002 Outsource Laboratories.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Outsource Laboratories, Inc. is independent of Sun Microsystems, Inc.

Table of Contents

About Olabs' Training Manuals	xiii
---	------

1	Basics of Programming	1-1
	Introduction	1-1
	What is a Program?	1-1
	Steps for Developing a Program	1-4
	Identify a Problem	1-4
	Outline the Solution by Writing an Algorithm.	1-4
	Implement the Algorithm using a Programming Language	1-5
	Run the Program.	1-5
	Maintain the Program.	1-5
	Introduction to Pseudocode	1-6
	Taking Input	1-7
	Providing Output	1-7
	Performing Arithmetic	1-8
	Storing a Value	1-8
	Choosing between Alternative Actions	1-8
	Repeating Actions	1-9
	Example Programs in Pseudocode	1-11
	Example 1: Conversion of Fahrenheit to Celsius	1-11
	Example 2: Calculate Total Including Sales Tax	1-11
	Example 3: Provide the Name of the Month	1-12
	What does the Computer do with a Program?	1-14
	Machine Instructions	1-14
	Compilers	1-15
	Compiling and Running a Java Program	1-18
	Using the DOS Prompt.	1-18
	Changing into a Different Folder	1-19
	Listing All Files in a Folder	1-19
	A few Words About Files and Filenames.	1-21
	Compiling a Java Program	1-22
	Running a Java Program.	1-23
	Exercise 1-1: Compiling and Running Java Programs	1-25
	Review Questions	1-27

2	Using Data Types, Variables and Operators	2-1
	Introduction	2-1
	The Structure of a Java Program	2-2

Running the Program	2-4
Statements	2-6
What are Variables ?	2-7
Declaring Variables.....	2-7
Assigning Values to Variables.....	2-9
Variable Names.....	2-10
Exercise 2-1: Find Valid Variable Names	2-11
What are Data Types?	2-12
The Data Type integer	2-13
Exercise 2-2: Declare a Variable of Type integer	2-14
Exercise 2-3: Assigning Values to Variables of Type integer	2-16
The Data Type ‘double’	2-19
Exercise 2-4: Declare a Variable of Type double	2-21
Exercise 2-5: Assigning Values to Variables of Type double	2-23
The Data Type ‘boolean’	2-25
Exercise 2-6: Declare a Variable of Type boolean	2-27
Exercise 2-7: Assigning a Value to a Variable of Type boolean	2-28
The Data Type character	2-29
Exercise 2-8: Declare a Variable of Type character	2-31
Exercise 2-9: Assigning Values to Variables of Type character	2-33
Exercise 2-10: Identify Values of Type character	2-35
Characters and their associated number code	2-36
The Data Type ‘String’	2-37
Shortcut for Printing Strings.....	2-38
Assigning a Value when Declaring a Variable	2-39
Exercise 2-11: Using String Variables	2-40
Exercise 2-12: Writing a Program	2-42
Exercise 2-13: Changing the String Value	2-43
Exercise 2-14: Map Values to Variables	2-44
Operators	2-45
Addition Operator (+)	2-47
Subtraction Operator (-)	2-49
Multiplication Operator (*)	2-50
Division Operator (/)	2-52
Exercise 2-15: Using Operators	2-54
Exercise 2-16: CoffeeHouseProgram	2-57
Exercise 2-17: Calculating the Average Grade	2-58
Exercise 2-18: Calculate the Weekly Salary	2-59
Modulo Operator (%)	2-60
Adding Strings (+)	2-62
Order of Operations - Precedence of Operators	2-64
Exercise 2-19: Change the Order of Operations	2-66
Comments in Java	2-67
Exercise 2-20: Using Comments	2-69
Exercise 2-21: The main Method	2-70
Unit Review Questions	2-71

3	Flow Control	3-1
	Introduction	3-1
	What is Flow Control?	3-2
	Selection Statements	3-3
	Comparison Operators	3-4
	Exercise 3-1: Using Comparison Operators	3-6
	Types of if-statements	3-7
	The if/else statement	3-7
	If-statement without else	3-12
	Multiple if-statements	3-14
	Exercise 3-2: Understanding Multiple If-Statements	3-18
	Breakfast Program as an Example for a multiple if-statement	3-19
	Exercise 3-3: Using an If-Statement	3-22
	Exercise 3-4: Making Decisions	3-24
	Exercise 3-5: Using a Multiple If-Statement	3-25
	Exercise 3-6: Making More Decisions	3-26
	The Switch Statement	3-27
	Exercise 3-7: Using a Switch Statement	3-32
	Repetition Statements	3-33
	While Loop	3-34
	Exercise 3-8: Keeping Track of Variable Values	3-39
	Exercise 3-9: Using a While Loop I	3-43
	Exercise 3-10: Using a While Loop II	3-44
	Exercise 3-11: Do While Loop	3-46
	Exercise 3-12: Reading Grades in a Loop (optional)	3-47
	Do while Loop	3-48
	Exercise 3-13: Using a Do While Loop	3-53
	Another Repetition Statement - for loop	3-55
	Incrementing and Decrementing Counter Variables	3-56
	Exercise 3-14: Incrementing and Decrementing Variables	3-60
	The Tomato Soup Recipe as a Program	3-61
	Review Questions	3-64

4	Using Arrays	4-1
	Introduction	4-1
	Introducing Arrays	4-2
	Creating an Array	4-5
	Creating an Array - Syntax I	4-5
	Exercise 4-1: Creating an Array	4-7
	Creating an Array - Syntax II	4-8
	Accessing Array Elements	4-11
	Retrieving an Array Element	4-11
	Changing an Array Element	4-13
	Exercise 4-2: Identifying Array Elements	4-14
	Retrieving all Array Elements	4-16

Print all Array Elements to the Screen	4-16
The length Property	4-20
Example Programs Using Arrays.	4-21
Example I: Representing Rows of People in a Theater	4-21
Example II - Weekday Translation Program.	4-23
Exercise 4-3: Accessing Elements of an Array of Strings	4-26
Example III - Array of Prices - Summing Up.	4-27
Example IV - Array of Prices - Finding the Lowest Price	4-29
Example V - Finding your Destination.	4-31
Exercise 4-4: Find Your Destination	4-33
Review Questions	4-34
5 Functions	5-1
Introduction	5-1
What is a Method?	5-2
Syntax for Defining a Method	5-4
A Method - as General as Possible	5-5
Another Method from the Cooking Application.	5-8
Two Types of Methods	5-10
Method with Return Type.	5-10
Exercise 5-1: Identifying Parts of a Method	5-13
Exercise 5-2: Calling Methods I	5-14
Using Methods inside a Statement as a Value	5-14
Exercise 5-3: Distinguishing Different Types of Methods	5-16
Example Programs Using Functions	5-17
Example 1 - CostCalculator	5-17
Example 2 - Convert Inches to Centimeters.	5-19
Example 3 - Greeting Program	5-21
Exercise 5-4: Calling Methods II	5-23
Exercise 5-5: Converting Currencies	5-24
Review Questions	5-25
6 Introduction to OO Programming	6-1
Introduction	6-1
Procedural vs. Object Oriented Programming.	6-2
Relationship between Classes and Objects	6-3
Exercise 6-1: Creating Objects from Classes	6-7
Encapsulation.	6-8
Exercise 6-2: Defining Attributes and Methods	6-10
Constructor.	6-11
Inheritance Relationships.	6-13
Review Questions	6-15

7	Java Programming Environment	7-1
	Introduction	7-1
	How Does a Computer Work?	7-2
	Data	7-2
	Operations	7-3
	Statements	7-5
	Compilers	7-6
	Programming and Programming Languages	7-7
	The Java Programming Language	7-8
	Java Virtual Machines	7-9
	The Java Development Process	7-10
	Java Development Tools	7-11
	Java Development Tool: javac	7-11
	Java Development Tool: java	7-13
	Exercise 7-1: Compile and Execute a Java Program	7-15
	Kinds of Java Applications	7-16
	Console Applications	7-16
	GUI Applications	7-17
	Servlets	7-18
	Applets	7-19
	Embedded Applications	7-20
	Java Virtual Machines Revisited	7-21
	Java Features	7-22
	Exercise 7-2: Create Your First Java Application	7-23
	Review Questions	7-25

8	Using Statements, Variables, Types and Values	8-1
	Introduction	8-1
	How Programs Work	8-1
	The Statement Terminator “;”	8-3
	Introducing Variables and Assignment	8-4
	Variables	8-4
	The Assignment Operator	8-4
	The Need for Variable Declaration	8-6
	Example	8-7
	Primitive Data Types	8-8
	Declaring Variables	8-9
	Exercise 8-1: Declaring a Variable	8-12
	Exercise 8-2: Performing Arithmetic	8-13
	Variable Initialization	8-14
	Literals	8-15
	Operators	8-16
	Exercise 8-3: Concatenating Strings	8-20
	Type Errors	8-21
	Type Conversion	8-22

Type Casting	8-24
Converting non-String values into Strings	8-26
Comments and Commenting	8-27
Review Questions	8-29
Exercise 8-4: Using Variables (optional)	8-31
9 Using Methods and Flow Control	9-1
Introduction	9-1
Introducing Methods	9-1
Types of Methods	9-5
Where Do Methods Come From?	9-6
Introducing the java.lang.Math Class	9-9
Exercise 9-1: Random Numbers	9-12
Introducing the java.lang.System Class	9-13
The LearningIO Class	9-13
Using Standard Java I/O	9-17
Partial Member Listing for the LearningIO Class	9-18
Exercise 9-2: Using Learning IO	9-19
if Statements	9-20
More about if statements	9-22
Statement Blocks	9-23
Exercise 9-3: Using if Statements	9-24
while Statements	9-25
Exercise 9-4: Using while Statements	9-28
Initializer Statements	9-29
Incrementer/Decrementer Statements	9-30
Infinite Loop Example	9-32
for Statements	9-33
About the ‘++’ and ‘--’ operators	9-34
for Loop with Statement Block	9-35
for Statement Summary	9-37
Exercise 9-5: Using for Statements	9-38
Exercise 9-6: Combining for and if Statements	9-39
Review Questions	9-41
Exercise 9-7: Using Loops (optional)	9-42
10 Using Objects	10-1
Introduction	10-1
Introducing Objects	10-1
Why Are Objects Important?	10-2
Where do Objects Come From?	10-2
A World of Objects	10-4
Object Variables	10-6
Declaring Object Variables	10-7
Creating Objects with the new Operator	10-7

The Constructor	10-7
Accessing Object Attributes with the “.” Operator	10-9
Access Modifiers	10-10
Exercise 10-1: Creating and Using Objects	10-11
Calling Object Methods with the “.” Operator	10-12
Member Signatures	10-12
Using the Java API Documentation.	10-14
Initializing Objects with Constructors.	10-15
Person Class Members.	10-16
Method and Constructor Overloading	10-17
Exercise 10-2: Using the Person Class	10-19
The java.lang.String Class	10-20
Constructor Signatures for Strings	10-21
Method Signatures of the java.lang. String class	10-22
Exercise 10-3: Using the java.lang.String class	10-24
Assignment, Multiple Aliases and Sharing	10-25
Side Effects of Multiple Aliases	10-26
Example: Sharing Objects May Cause Side Effects	10-27
Garbage Collection	10-28
The null Reference.	10-29
Dereferencing a null Reference	10-30
Java is All About Objects!	10-32
The Database Class	10-32
Overview of the CD Database System Exercise	10-34
Exercise 4-4: The CD Database System (Part 1).	10-35
Exercise 4-4: The CD Database System (Part 2).	10-35
Exercise 4-4: The CD Database System (Part 3)	10-36
Exercise 4-4: The CD Database System (Part 4).	10-36
Exercise 4-4: The CD Database System (Part 5).	10-37
Review Questions	10-39
Exercise 10-5: Using Strings (optional)	10-40

11 Introducing Arrays 11-1

Introduction	11-1
Why Arrays Are Needed	11-1
Introducing Arrays.	11-5
Declaring Array Variables	11-6
Creating Arrays with the new Operator.	11-7
The [] Operator - Assigning a Value to an Element	11-8
The [] Operator - Retrieving the Value of an Element	11-9
Common Problem: Array Index Out Of Bounds.	11-10
Determining the Length of an Array	11-11
Initialization at Declaration	11-12
Scenario: Using Variables/Expressions as Indices	11-13
Scenario: Initializing Array Elements	11-14

Scenario: Output Array Elements	11-15
Scenario: Reading Array Elements	11-16
Scenario: Summing Array Elements	11-17
Exercise 11-1: Calculating the Average	11-18
Scenario: Counting Array Elements	11-19
Scenario: Filtering Array Elements	11-21
Scenario: Searching for Array Elements	11-22
Scenario: Locating the Minimum/Maximum Elements.....	11-24
Scenario: Sorting Array Elements	11-25
Application Command Line Parameters	11-26
Exercise 11-2: Fibonacci Numbers	11-28
Exercise 11-3: Object Arrays (optional)	11-29
Review Questions	11-31
Exercise 11-4: Using Arrays (optional)	11-32

12 Developing Classes

12-1

Introduction	12-1
Class Construction Checklist	12-2
Case Study: A Banking Application.....	12-2
Step 1: Name the Class	12-4
A Minimal Class Definition	12-4
The main Method 12-7	
Exercise 12-1: A Minimally-Defined Class	12-8
Step 2: Determine and Declare the Attributes	12-9
Exercise 12-2: Class Attributes	12-12
Step 3: Determine, define, and implement the Methods	12-14
Step 3a: Defining Methods - Determine Operations.....	12-14
Step 3b: Defining Methods - Define Signatures and Return Types	12-14
Naming Conventions for Java Methods 12-15	
Case Study Example: Determine and Define the Methods 12-17	
Step 3c: Implement the Methods	12-19
How Method Calls Work 12-20	
Implementing the deposit and withdraw Methods 12-21	
Implementing the calculateInterest method 12-22	
The return Statement 12-22	
Local Variable Declarations 12-23	
Methods Implemented in BankAccount Class	12-23
Exercise 12-3: Implementing Methods	12-28
Initializing Objects: Constructors Revisited	12-29
Creating a No-Argument Constructor.....	12-30
Overloaded Constructors	12-31
Exercise 12-4: Define a Constructor	12-32
Introducing Inheritance	12-34
The extends Keyword	12-35
Using Inheritance	12-36
The super Reference	12-38
Polymorphism.....	12-39
Access Modifiers: private, public and protected	12-41

The static Modifier	12-45
Exercise 12-5: Using a static Attribute	12-46
Exercise 12-6: A Banking Application	12-47
Review Questions	12-53
Optional Examples	12-55
Exercise 12-7: Using Arrays and Methods (optional)	12-55

13 Data Structures	13-1
Introduction	13-1
What are Data Structures?	13-1
Java Data Structures	13-3
The Vector Class	13-4
The Object Class.	13-4
Selected Methods from the Vector class	13-5
Vector Class Example	13-6
Exercise 13-1: Using the Vector Class	13-7
Wrapper Classes	13-8
Wrapper Class Example.	13-9
Exercise 13-2: Using the Wrapper Class	13-10
The Hashtable Class	13-11
Exercise 13-3: Using the Hashtable Class	13-13
Enumerating Values of a Data Structure	13-14
A Brief Introduction to Interface Classes.	13-14
Enumeration Interface Example.	13-15
Exercise 13-4: Using Vectors and Wrapper Classes	13-17
Review Questions	13-19

14 Where to From Here?	14-1
Introduction	14-1
do-while Statements.	14-1
The switch Statement.	14-3
Exercise 14-1: Using a switch Statement	14-5
Method Call Chaining	14-6
Introducing Exceptions	14-7
Exceptions Example - Without Exception Handling	14-7
How Java Handles Exceptions	14-9
Exceptions Example - Using Exception Handling.	14-9
Java's Catch or Specify Requirement.	14-11
Checked vs. Unchecked Exceptions 14-11	
Example Exception Handlers	14-12
The try block.	14-13
The catch block(s)	14-14
The finally block.	14-16
Specifying Exceptions Thrown by a Method.	14-16
Important Java Classes and Concepts	14-17

Java Input and Output	14-17
Threads and Multithreading	14-17
Graphical and Internet Applications	14-18
 A Java's Reserved Keywords, Operators and Flowchart Symbols	 A-1
Java's Reserved Keywords.	A-1
Operators and their Precedence	A-2
Flowchart Symbols	A-3
 Index	 I-1

About Olabs' Training Manuals

This training manual is organized into units, and within a unit it is organized by up to three levels of headings. The unit title and first level heading are shown in top header of each page following the first page of a unit. This allows you to quickly relate a second or third level heading that occurs in a page to the main topic it is related to by simply looking at the page header.

The figures in the manual coordinate with presentation files used by the instructor. Thus, for example, if the instructor is showing a slide labeled Figure 2-20, you will be able to find this in your manual labeled with the same number (the figure number 2-20 references the 20th figure in Unit 2).

Most units conclude with review questions. Answers to these questions can be found in an appendix of the manual.

Exercises are placed throughout the units, close to relevant topics. Exercises may require coding, however some require analysis only. Icons next to the Exercise heading indicate what type of exercise it is. Icons are used in other situations within the manuals as well.

The following icons are used in the training manuals, however each one may not be used in every manual:



This icon represents a hands-on exercise that requires compiling a program and running it.



This icon represents a hands-on exercise which is done on paper or as a discussion; it does not require editing, compiling or running a program.



This icon represents a running code sample available on line that is to be studied.



This icon represents that the material is related to a sample application.

Accompanying this manual is a set of electronic files which contain exercise templates and solutions organized to follow the unit structure of the manual, sample programs referenced in the manual, as well as solutions to analysis type exercises. These files are supplied to you by the training organization. Third-party software required for the course is not distributed by Outsource Laboratories.

1 Basics of Programming

Introduction

In this unit we begin by providing some background about what computer programs do and what the process of programming is about. Then we introduce you to pseudocode, an English language like way to document the steps of a program. The Java programming environment is introduced and we will learn how to run a Java program. The intended learning outcomes for this unit are listed in Figure 1-1.

At the end of this unit, the student will be able to:

- Identify programming problems.
- List the steps in developing a program.
- Identify actions expressed in pseudocode.
- Compile and run a Java program.

Figure 1-1: Unit Objectives

What is a Program?

A computer helps us to automate everyday processes. A program runs on a computer. A program is a sequence of steps that the computer executes.

The computer itself is dumb, a computer can't come up with a problem and find a solution. Humans have problems that they want to have solved by a computer, because they are repetitive and boring to solve or they are complex and would take too long to solve by humans.

Computers are very good at making calculations for us according to a given sequence of steps or formula. In a program therefore we define a sequence of steps that we want the computer to execute according to a problem that we want to have solved.

Examples of problems that could be solved using a computer:

Example 1:

Your friend from Europe came to visit you in America. You are watching the weather forecast. The temperature is given in Fahrenheit, but your friend is used to temperature measures in Celsius. The problem is you have to convert the Fahrenheit temperature into a Celsius temperature so your friend knows what sort of weather to expect. The formula to convert Fahrenheit into Celsius is: $\text{Celsius} = 0.55 * (\text{Fahrenheit} - 32)$.

You could write a computer program to do this.

Typically a program has three main steps as listed in Figure 1-2.

The three main parts of every program

1. Take input.
2. Process the input.
3. Provide output.

Figure 1-2: Three Main Parts of a Program

In the case of the temperature conversion program the input would be the temperature in Fahrenheit, the processing is the application of the formula to convert the Fahrenheit value into a Celsius value. The output is the temperature in Celsius.

Given this problem you could write a program that fulfills exactly these steps and could be run every time you need to convert a Fahrenheit temperature into a Celsius temperature.

Programs would normally be designed to be general, meaning they are not built for just one particular problem, but for many problems that are similar.

Example 2:

Another problem that could be solved using a computer is the following:

While shopping you want to add up the prices of the goods you intend to buy and also calculate the total including the tax.

In terms of the three typical steps every program has for this program there are:

Input: Take in prices of goods that you are buying.

Process: Calculate the sum of the prices, calculate the tax you need to pay on that sum and add tax to sum of prices.

Output: The total cost including sales tax.

Example 3:

Another program could provide you with the name of the month given a number between 1 and 12.

Input: A number digit between 1 and 12.

Process: Compare the number that was given with a list of numbers in the program that have names corresponding to the number. Find the respective name of the month.

Output: Provide the name of the month.

Thus, this program, if given the digit 5, would print “May” to the screen or send it to a printer.

The three example programs are listed in Figure 1-3.

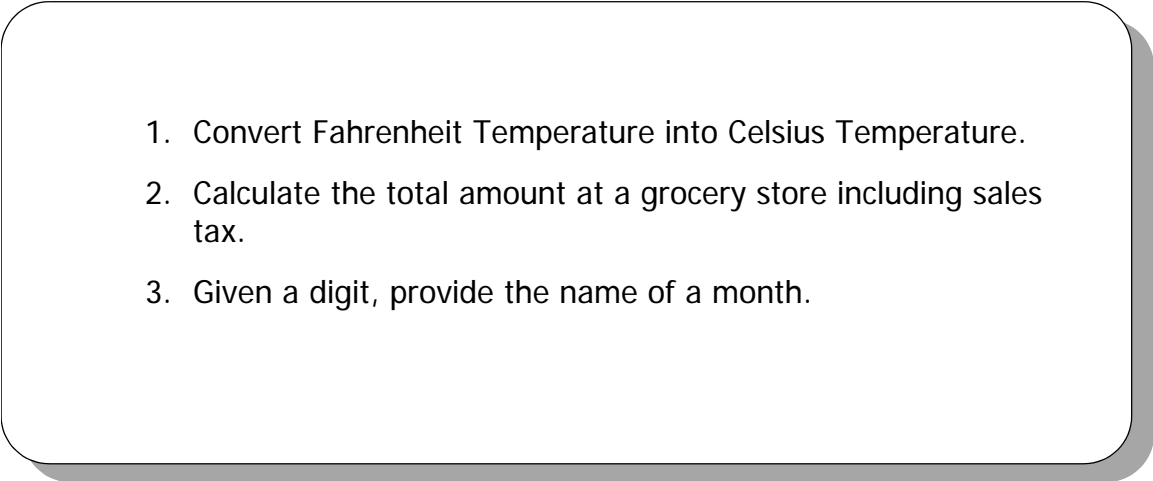
- 
1. Convert Fahrenheit Temperature into Celsius Temperature.
 2. Calculate the total amount at a grocery store including sales tax.
 3. Given a digit, provide the name of a month.

Figure 1-3: Examples of Problems for Programs

Steps for Developing a Program

The design process of a program always starts with a problem. We just discussed three example programs. At the start of each was a problem that needed solving. Before you can finally run the program there are other steps involved.

In Figure 1-4 the steps that you would normally take in designing and writing a computer program are listed.

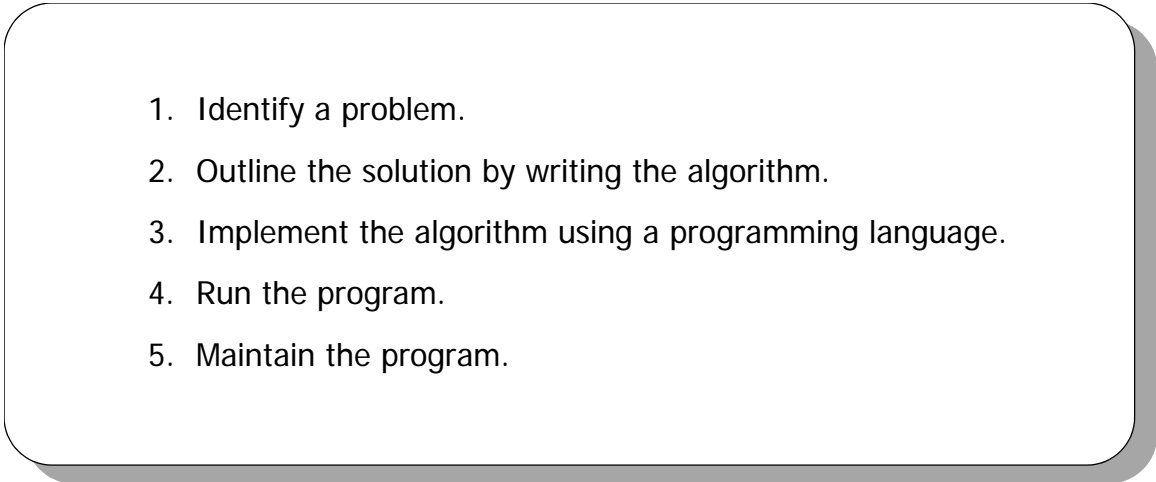
- 
1. Identify a problem.
 2. Outline the solution by writing the algorithm.
 3. Implement the algorithm using a programming language.
 4. Run the program.
 5. Maintain the program.

Figure 1-4: Steps for Developing a Program

Let's discuss each of the steps.

IDENTIFY A PROBLEM

First you need a problem that you want to solve. It helps to define the problem in great detail thinking of a general way to formulate it.

OUTLINE THE SOLUTION BY WRITING AN ALGORITHM

In this step you need to understand the problem and find a solution for the problem.

To outline the solution you can write down the steps the program needs to take to produce the desired result.

A program is similar to a recipe to prepare a dish. In a recipe, for example a recipe to prepare a soup, you list the steps that you need to take to be able to enjoy the soup. The steps have to be taken in a given sequence. You can't take a step that is at the end of the sequence for example the garnishing of the soup before you have even cut and cooked the vegetables that you prepare the soup with.

In a program, similar to the recipe you list the steps the program needs to take to produce the result. This sequence of steps is called an algorithm.

An algorithm should be formulated as close to the programming language as possible. In this course we will list the steps for your program using so-called pseudocode. We will teach you how to document the steps for your programs using pseudocode.

We need to be very specific and concise when giving tasks to the computer. As you will see the pseudocode that we will use is already very restricted in use of words and symbols.

Having our steps listed in an algorithm we can test it with example input values.

IMPLEMENT THE ALGORITHM USING A PROGRAMMING LANGUAGE

The outline of the solution in an algorithm is still very general. We could use this outline to implement the program in any programming language. So really once you have defined an algorithm for a program you have to choose which programming language to use.

In this course we are using Java. Each programming language has its own set of rules, symbols and special words. Learning a programming language is a lot about learning these rules and symbols and how they are used in a program.

Once an algorithm is translated into a programming language you call it code. The process of writing a program in a programming language is called coding. A big part of programming however is the part of coming up with the algorithm.

RUN THE PROGRAM

Depending on the programming language you are using, the way you run the program might be different. In this unit you will learn how to run a Java program.

MAINTAIN THE PROGRAM

Most programs are written to be used not just once but many times. Over time maintenance is usually required. When you are dealing with more complex problems, changes to the code might be necessary after a while or there are simply some bugs in the program because not every possible test has been performed on the program.

It is also necessary to comment your program. When you write a complex program and try to understand and edit it after a year or so you most likely don't know anymore what you were thinking when you wrote the program first. For this case, and for the more likely case that another programmer needs to understand your code, you have to incorporate comments into your code that explain what your program does.

We have already discussed three examples of how to define a problem for a program. To take the next step, to outline the solution using an algorithm we need to learn how to write the so-called pseudocode that we are using in this course.

Introduction to Pseudocode

In this course we will use pseudocode to document the algorithm of a program. There is no one standard for pseudocode. It is a way to formulate the programming problem using prose which is as close as possible to the way you write programs in the programming language, without using any of a programming language specific syntax.

We now introduce you to pseudocode as we will use it in this book. Then we will look at our example programming problems again and formulate them in pseudocode. At the end of this unit you should be able to read pseudocode to understand the logic of a program. You won't have to write pseudocode yourself in this course.

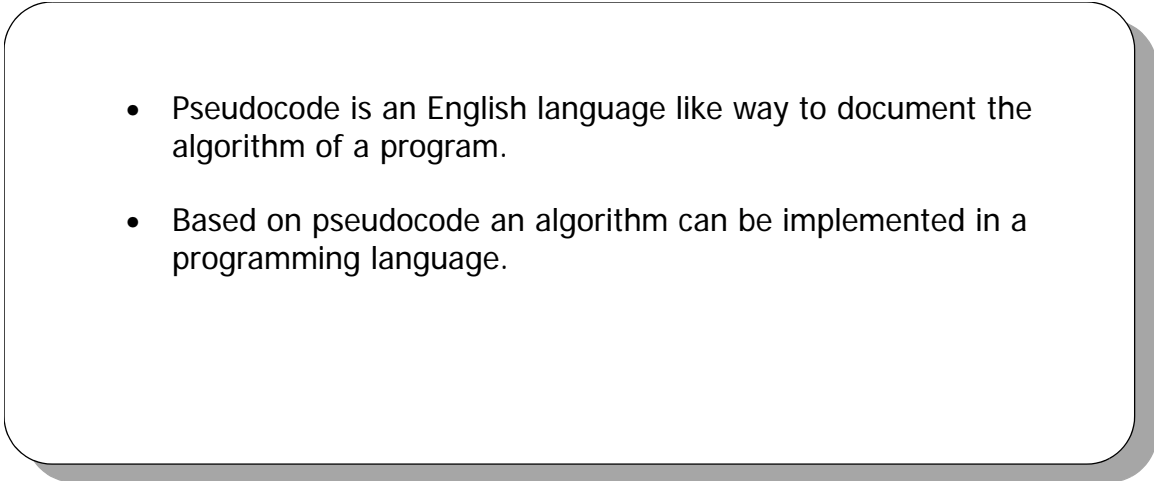
- 
- Pseudocode is an English language like way to document the algorithm of a program.
 - Based on pseudocode an algorithm can be implemented in a programming language.

Figure 1-5: Introduction to Pseudocode

The pseudocode is a step in between normal language and the language the computer understands to get us slowly used to the technical terms of the programming language.

To solve most of the programming problems we use six basic actions that are listed in Figure 1-6. We will look at each of these in more detail.

1. Taking input.
2. Providing output.
3. Performing arithmetic.
4. Storing a value.
5. Choosing between alternative actions.
6. Repeating an action or set of actions.

Figure 1-6: Basic Actions in a Program

TAKING INPUT

Taking input describes the event when the program receives information over the keyboard or another device. In pseudocode we describe any receiving of input with the word 'read' .

The pseudocode for the part of a program where we read in data, for example the price of an item in a shop, would be written as follows:

`Read price`

How would you write the pseudocode for the part of a program where the number of items being purchased is input?

PROVIDING OUTPUT

Providing output describes the actions when the program provides information either through the printer or on the screen.

We use the word 'print' in pseudocode to indicate that information is provided.

The pseudocode for the part of the program that describes the output for the sum of prices for items that you bought is as follows:

`Print sum`

How would you write the pseudocode for the part of the program where the converted temperature is provided.

Example Programs in Pseudocode

Now let us look at the pseudocode of the three example programs we discussed above.

EXAMPLE 1: CONVERSION OF FAHRENHEIT TO CELSIUS

Let's outline a solution for our first example program, the conversion from Fahrenheit to Celsius. We have to perform the following steps:

1. Take in the temperature in Fahrenheit
2. Apply the formula to convert a temperature from Fahrenheit to Celsius.
3. Print out the temperature in Celsius.

In pseudocode the program can be described as follows:

```
Converter  
  Read temperature in Fahrenheit  
  Apply formula to convert temperature  
  Print temperature in Celsius  
END
```

Figure 1-8: Pseudocode for Temperature Converter

Converter is the name of the program. In this example we use the special word 'read' to describe the process of reading in a temperature value. In the second line we apply the formula to convert Fahrenheit into Celsius and assign the resulting value to a variable called celsius.

As a result of running the program, the calculated celsius temperature is printed to the screen.

EXAMPLE 2: CALCULATE TOTAL INCLUDING SALES TAX

An outline for a solution for our second problem - calculate the total including sales tax of goods that we buy at a store - is as follows:

1. Read the prices.
2. Calculate the sum.
3. Use the sum to calculate the tax.
4. Calculate the total sum with tax.
5. Output the grand total.

In pseudocode the program can be described as follows:

```
TotalCalculator
  Read the prices.
  sum = price1 + price2 + price3
  tax = ((sum * 8) / 100)
  total = sum + tax
  Print out total
END
```

Figure 1-9: Pseudocode for Total Calculator

First we need to read in the prices of the good we intend to buy, then we calculate the sum of the prices by adding the prices up. Next we calculate the tax (in this case 8%) for all goods and add it to the sum. Then we can print out the total amount we have to pay.

EXAMPLE 3: PROVIDE THE NAME OF THE MONTH

And let's do the same thing for the third problem we described.

In pseudocode the algorithm can be described as follows:

```
Months
  Read month
  IF month == 1
    print 'January'
  ELSE IF month == 2
    print 'February'
  ELSE IF month == 3
    print 'March'
  ELSE IF month == 4
    print 'April'
  ELSE IF month == 5
    print 'May'
  ELSE IF month == 6
    print 'June'
  ELSE IF month == 7
    print 'July'
  ELSE IF month == 8
    print 'August'
  ELSE IF month == 9
    print 'September'
  ELSE IF month == 10
    print 'October'
  ELSE IF month == 11
    print 'November'
  ELSE IF month == 12
    print 'December'
END
```

Figure 1-10: Pseudocode for Month Name Finder

In this program we first read in the digit representing the month, then we compare this digit with the digits from 1 to 12. Once we find the exact digit we print out the respective name of the month.

Compiling and Running a Java Program

USING THE DOS PROMPT

Throughout this course we will be using the so-called DOS or command prompt to compile and run our programs. Through the command prompt you can talk directly to the DOS operating system. In the following section we will introduce you to some of the commands that you will use when dealing with Java programs.

Starting a Command Prompt Window

First we have to start the command prompt window. Assuming you are working on a Windows machine you should have a Start button in the left bottom corner. Click on the 'Start' button and choose 'Programs' as the option. Then choose 'Command Prompt' as the option. A new window should appear on your screen that looks as shown in Figure 1-15.

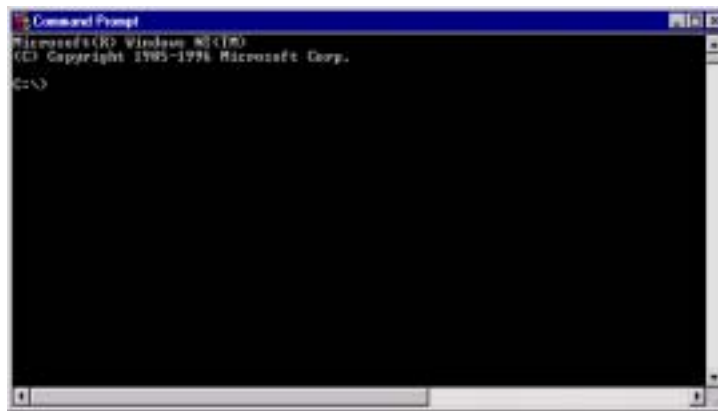


Figure 1-15: Starting a Command Prompt

Start a new command prompt on your computer now.

When the Command Prompt window first opens it always shows the system prompt `c:\`

The hard disk of your computer is called `c` in most cases. On your disk you can arrange a hierarchy of directories. When you first start the command prompt it directs you to the main disk. From there you can wander through the folder hierarchy.

You have probably worked with Windows Explorer and have seen how you can structure your drives on the computer into different directories. In the Windows Explorer you could click on a folder that represents a folder to see the contents of that folder.

When using the Command Prompt we have to use commands to change to a different folder and list all files that are contained in a folder or to run programs on files.

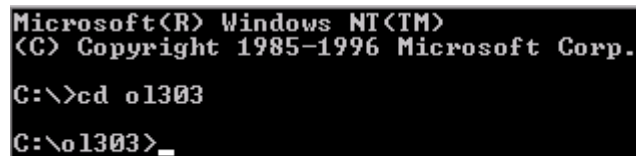
All the exercise files for this course are in a folder called ol303.

So we should first learn how we can get to this folder from the c:\ prompt.

CHANGING INTO A DIFFERENT FOLDER

The command to change a folder is: cd (change **d**irectory).

To change into the folder ol303 that stores all lab files for this course you should type the following command at the command prompt: cd ol303. This is shown in Figure 1-16.



```
Microsoft(R) Windows NT(TM)  
(C) Copyright 1985-1996 Microsoft Corp.  
C:\>cd ol303  
C:\ol303>_
```

Figure 1-16: Changing into a Different Folder

At your computer change into the folder ol303 or if this folder does not exist into the folder that holds the exercise and example files for this course.

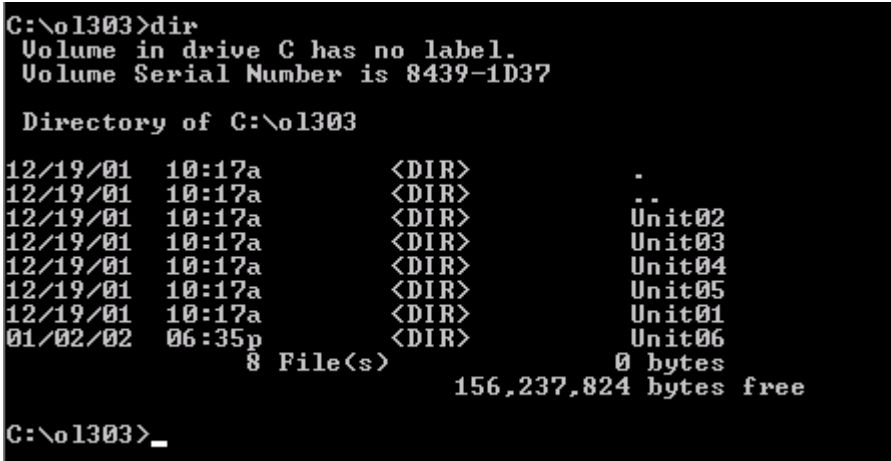
Once you have typed in the command 'cd' followed by the name of the folder you want to change into, the prompt indicates that you are now in the folder with the name ol303.

LISTING ALL FILES IN A FOLDER

Now a good thing to do would be to list all the files that are contained in this folder. There is the command 'dir' that will list all files and subdirectories in the current folder.

Type the command 'dir' at the command prompt and see what happens.

The result is shown in Figure 1-17.



```
C:\ol303>dir
Volume in drive C has no label.
Volume Serial Number is 8439-1D37

Directory of C:\ol303

12/19/01  10:17a    <DIR>          .
12/19/01  10:17a    <DIR>          ..
12/19/01  10:17a    <DIR>          Unit02
12/19/01  10:17a    <DIR>          Unit03
12/19/01  10:17a    <DIR>          Unit04
12/19/01  10:17a    <DIR>          Unit05
12/19/01  10:17a    <DIR>          Unit06
01/02/02  06:35p    <DIR>          Unit06
               8 File(s)                0 bytes
               156,237,824 bytes free

C:\ol303>_
```

Figure 1-17: Listing all Files

On the screen, 6 sub folders are displayed, one folder for every unit of the course. Each of these sub folders contains the exercise and example files for that unit.

On your computer list all files that are in the folder that you are currently in by using the command 'dir'.

To get into one of those sub folders, what do you think you have to do?

Yes, change the folder again. How would you do that?

Type in the command 'cd' followed by the name of the folder that you want to change into: e.g., 'cd Unit01'

Please change now into the folder 'Unit01' and then list all files that are contained in that folder as shown in Figure 1-18.

```
C:\o1303\UNIT01>dir
Volume in drive C has no label.
Volume Serial Number is 8439-1D37

Directory of C:\o1303\UNIT01

12/19/01  10:17a      <DIR>          .
12/19/01  10:17a      <DIR>          ..
01/02/02  07:01p      755 Converter.class
01/02/02  11:46a      743 CostCalculator.class
01/02/02  11:47a      872 CostCalculator.java
01/02/02  11:47a      1,413 Months.java
01/02/02  11:48a      899 Months.class
01/07/02  12:18p      973 Converter.java
           8 File(s)          5,655 bytes
           187,432,960 bytes free

C:\o1303\UNIT01>
```

Figure 1-18: Listing all Files in Folder Unit01

If you want to be able to go back to the folder above the subfolder you are in:

```
cd ..
```

You can always use 'dir' to check which files are contained in the folder you are in.

A FEW WORDS ABOUT FILES AND FILENAMES

When you list the files inside one of the units you see that they have a name, followed by a dot followed by either java or class. Each filename is made up of two parts: the file name (the piece before the dot) and the extension (the piece after the dot). The extension tells you of what type the file is. All files with the extension .java are files with Java code (like the ones you'll be writing), files with the extension .class are compiled Java programs created from the .java file. You might have seen other extensions such as .txt for a text file or .doc for a text file that has been created using Win word. The parts of a file name are shown in Figure 1-19.

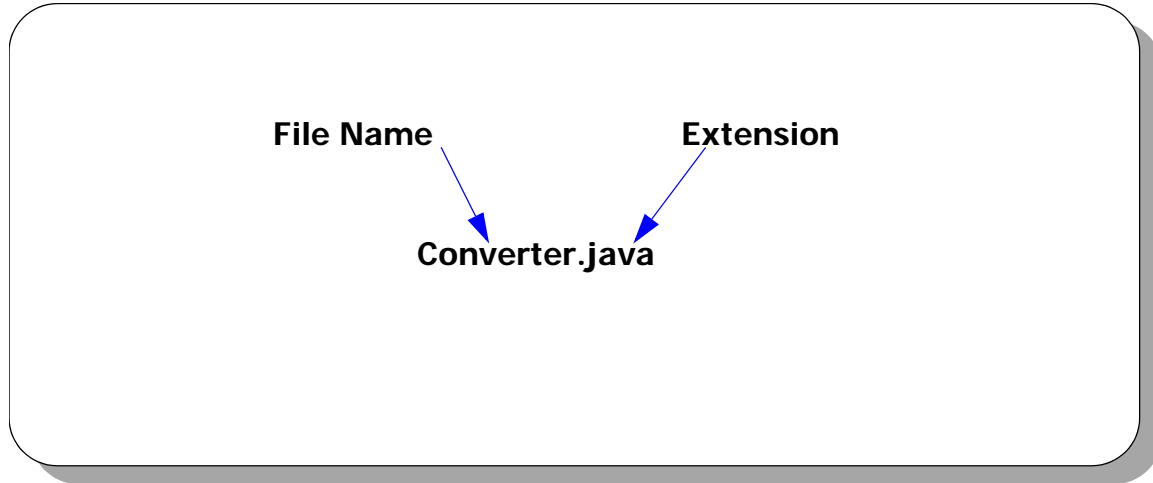


Figure 1-19: Parts of the Name of a File

The extension of a file will always help you to identify the type of file or program you are dealing with.

COMPILING A JAVA PROGRAM

One thing we will do throughout this course is compile Java programs. So we should now learn how we can do that.

To compile a Java program from the command prompt we use the command `javac`. To compile a particular file with Java code you need to:

1. Go to the folder that contains the file with your Java program. (It should be a file with the extension `.java`)
2. Type the word '`javac`' followed by the name of the program including the extension `.java`. If your program is called `Converter.java` you would type in:

```
javac Converter.java
```

as shown in the screenshot below.

To compile a Java program, type the command '**javac**' followed by the **name of the program including the extension .java** at the command prompt:

```
C:\o1303\UNIT01>javac Converter.java
```

Figure 1-20: Compiling a Java Program

As a result of executing this command, a class file (the runnable Java program) is created called Converter.class.

Go into the folder Unit01 if you are not already there and type in the command to compile the Converter program as shown in Figure 1-20.

RUNNING A JAVA PROGRAM


To run the compiled program (that is the class file) we use the command 'java'. To run a particular Java program, you need to:

1. Go to the folder that contains the compiled Java program (the file with the extension .class).
2. Type the word 'java' followed by the name of the program, this time without the extension. If your program is called Converter you would type in:

```
java Converter
```

as shown in the screenshot in Figure 1-21.

To run a Java program type the command 'java' followed by the name of the program at the command prompt:



```
C:\o1303\UNIT01>java Converter
80.0 in Fahrenheit are 26 degress in Celsius
C:\o1303\UNIT01>
```

Figure 1-21: Running a Java Program

Run the Converter program on your computer by typing in 'java Converter' as shown in Figure 1-21.

Exercise 1-1: Compiling and Running Java Programs



Purpose: Learn to compile and run Java programs.

Instructions: In the last section we learned how to compile and run a Java program. We compiled and ran one of our three example programs. Let's now compile and run the other two programs. Follow the instructions in Figure 1-22 and Figure 1-23 to complete this exercise.

1. Open a new command prompt.
2. Go to the folder ol303/unit01 that contains the example programs for this unit using the command 'cd'.
3. Compile the program CostCalculator by typing in the command:

```
javac CostCalculator.java
```
4. Check if the file has been compiled by using the following command to find the CostCalculator.class file:

```
dir
```
5. Run the program CostCalculator by typing in the command:

```
java CostCalculator
```
6. What is the result of running this program?

Figure 1-22: "Compiling and Running Java Programs" Exercise

1. Stay in the same folder.
2. Now compile the program Months by typing in the command:

```
javac Months.java
```
3. Check if the file has been compiled by using the command:

```
dir
```
4. Run the program Months by typing in the command:

```
java Months
```
5. What is the result of running this program?

Figure 1-23: “Compiling and Running Java Programs” Exercise (cont’d)

Review Questions

1. What are the three main parts of most programs?
2. Following are the steps for developing a program. They are jumbled up. Bring them into the correct sequence.

Run the program.

Outline the solution (algorithm).

Identify a problem.

Maintain the program.

Implement the algorithm using a programming language.

3. What is pseudocode?
4. Following there are two commands. Mark which of the command compiles the program and which one runs the program.

```
java Greeting  
javac Greeting.java
```
5. What is the extension of a Java program that is not yet compiled?
6. What is the extension of the file that is created when you compile a Java program?
7. Which DOS command lists all files in the current folder?
8. What is the DOS command to change into a different folder?
9. Below there is a table with example programming problems on the left hand side and statements written in pseudocode on the right-hand side.
Match the problems to the pseudocode

Programming Problems	Pseudocode
Output the result of a calculation	Cookies = 15 WHILE (cookies > 0) DO Eat one cookie cookie - 1 ENDWHILE
Eat cookies until they are all gone given 15 cookies.	Print result

Input the name of a person	IF age < 21 Print "Sorry you can't enter this place" ELSE Print "Please come in" ENDIF
Store the price of an item you buy in a shop	price = 67.80
Check if the age is less than 21. If that is the case print "Sorry, you can't enter this place". If the value is equal or more than 21 print "Please come in"	result = 7 * 45
Multiply 7 buy 45 and store the result.	Read name

2 Using Data Types, Variables and Operators

Introduction

In this unit we learn the basic building blocks of a program. Let's say you wanted to write a program to do a simple calculation such as summing up two prices for goods that you bought (e.g. \$423.70 + \$45.90). A program that executes this calculation would have to be flexible to allow you to sum up any two prices. Therefore, we need placeholders for the two dollar amounts. In programming terminology we call these placeholders variables. When running the program you would store different values in these placeholders and then have the program calculate with the respective values.

The computer needs to know what sort of values you want to store inside a variable. You could have a program where you store dollar amounts in your variables as in our example or you might want to store words in variables. You need to indicate to the computer what types of data you are using. These are the data types of the variables in programming terminology.

Finally we need to discuss what we can do with the variables. In our program we sum up two prices. In other programs you might want to multiply values. We call these operations, and the symbol that indicates which type of operation we want to perform is called an operator. The '+' in our price example is an operator. Other operators are (-) for subtraction or (*) for multiplication.

In order to write a program that does a simple calculation you need to know how to use variables, data types and operators. That's what you are going to learn in this unit. At the end of the unit you will be able to write your own little program using these basic building blocks. The intended learning outcomes for this unit are listed in Figure 2-1.

At the end of this unit, the student will be able to:

- Use variables and operators in a program.
- Differentiate the five main data types in Java.
- Use comments to document a program.
- Write a simple Java program.

Figure 2-1: Unit Objectives

The Structure of a Java Program

In the last unit we compiled and ran a few simple Java programs. Let us now look at the structure and the main parts of a Java program in more detail.

Every Java program starts with the word 'class' followed by the name of the program, followed by an opening curly bracket (`{`). Look at the first line of program code in Figure 2-2. There we see the word 'class' followed by 'AddingNumbers' (the name of the program) and an opening curly bracket (`{`).

The name of the program is also the name of the file that we save it as. The program AddingNumbers is saved in the file AddingNumbers.java .

Curly brackets indicate a grouping of lines of code. In the extract of program code that is shown in Figure 2-2 everything that belongs to the class AddingNumbers is grouped inside an opening (`{`) and a closing (`}`) curly bracket.

Inside the class there is a 'main method'. Again everything inside this main method is grouped together by an opening and a closing curly bracket. We will learn what a method is in unit 5.

The line 'public static void main (String[] args)' starts the main method. You don't need to understand the meaning of these words yet. Just keep in mind that each Java program needs to have a main method that starts with the sequence of words as you see in line 3.

Basic Structure of a Java program:

```
1  class AddingNumbers {  
2  
3      public static void main(String[] args) {  
4  
5          [This is where the rest of the  
6          program code goes]  
7      }  
8  }
```

Annotations in the diagram:

- keyword to start the program**: points to the `class` keyword on line 1.
- name of the program**: points to `AddingNumbers` on line 1.
- main method**: a bracket on the right side of lines 3 through 7.
- end of the program**: points to the closing curly brace on line 8.

Figure 2-2: Structure of a Java Program

Let us now look at an example program that we will be using at many points throughout this unit.

Our example program sums up two numbers and prints out the result.

To fully understand everything in this program we need to learn how to use variables, data types and operators. That is what we will be doing in the following sections. At the end of this unit you will be able to write a simple program all by yourself.

The problem we are solving with this program is as follows: Declare three variables of type integer, assign numbers to two of those variables, add these two numbers together, assign the value to the third variable and print the result.

The program would be written in pseudocode as follows:

```
AddingNumbers  
    Read number1  
    Read number2  
    result = number1 + number2  
    print result  
END
```

The steps 'Read number1' and 'Read number2' in the pseudocode involve the declaration of variables and assigning values to the variables. In Figure 2-3 the complete Java program 'AddingNumbers' is shown. Concentrate on the structure of the program for the moment. Look at the program in Figure 2-3 and answer the following three questions:

1. What is the name of the program?
2. In which line does the main method start?
3. In which line does the main method end?

```
1  class AddingNumbers {  
2  
3  public static void main(String[] args) {  
4  
5      // Declare three variables of type integer  
6      int number1;  
7      int number2;  
8      int result;  
9  
10     // Initialize two of the variables  
11     // number1 represents the first value  
12     // number2 represents the second value  
13     number1 = 678;  
14     number2 = 345;  
15  
16     // Sum the two numbers and store the result in 'result'  
17     result = number1 + number2;  
18  
19     // Print out the result  
20     System.out.println("The result of adding number1 and  
                           number2 is " + result);  
21 }  
22 }
```

Figure 2-3: Structure of a Java Program (cont'd)

Inside the main method we find the code that is actually doing the things we want the program to do. You will learn everything you need to know to understand this central part of the program in this unit. Before we dive into it, let us first compile and run the program.

RUNNING THE PROGRAM

You have already compiled and run a few programs in the last unit. Remember the steps you need to take? To remind you, the steps are listed in Figure 2-4.

1. Compile the program. At the command prompt type:

```
javac AddingNumbers.java
```
2. Check if the class file AddingNumbers.class has been created by using the dos command 'dir'. At the command prompt type:

```
dir
```

If the file AddingNumbers.class is listed, the program compiled without errors.
3. Run the program. At the command prompt type:

```
java AddingNumbers
```

Figure 2-4: Running the Program

Below a screenshot shows the two commands and the result of running the program:



```
C:\o1303\UNIT02>javac AddingNumbers.java
C:\o1303\UNIT02>java AddingNumbers
The result of adding number1 and number2 is 1023
C:\o1303\UNIT02>
```


Statements

Before we start discussing other concepts let's first talk about statements.

What is a statement? A statement represents a sequence of hundreds or thousands of machine instructions. These statements are translated into machine instructions when we compile and run the program. Every program is more or less a list of statements. In the program `AddingNumbers.java` that we saw in Figure 2-3 every line that ends with a semicolon (;) represents a statement.

For example:

```
line 13: number 1 = 678;    or  
line 17: result = number1 + number2;
```

A program is put together out of many statements and there are different types of statements defined in the rules (syntax) of the programming language.

In this unit we will discuss the form of statements to declare variables and to do simple operations. In the following units we will move on to more complex statements that let you solve more complex problems in your program.

What are Variables ?

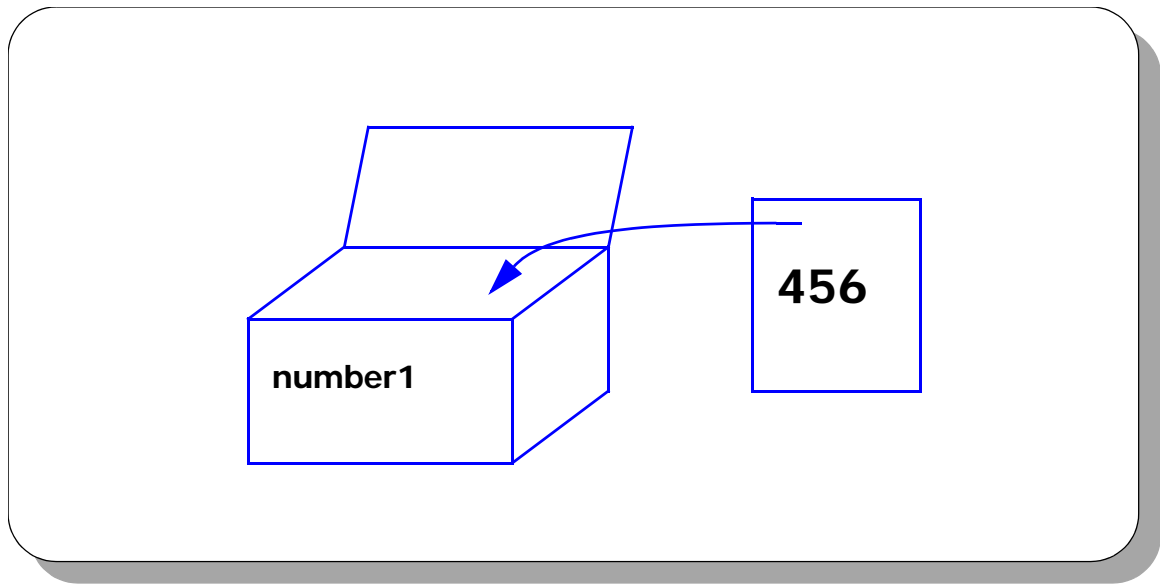


Figure 2-5: The Variable - A Storage Box for Values

Variables are like storage boxes for values. Imagine a variable as a little box with a name as shown in Figure 2-5. When we assign a value to that variable we can imagine that we put a piece of paper with a number or a word on it into the box. Each variable would be represented with one box. Whenever we use the name of the box in our program we have to open it and look at the number or word on the piece of paper inside the box. Then we can calculate with the numbers on the piece of paper in the box or write the number or word that is written on the piece of paper to the screen.

In the AddingNumbers program that we investigated we used three variables: number1, number2 and result. The variables number1 and number2 held two numbers and the variable result held the result of adding the two numbers.

DECLARING VARIABLES

When we want to use variables in a program we have to tell the program the names of our variables. In Java we call that the **declaration** of variables. In lines 6 to 8 in the program in Figure 2-3 we are **declaring** three variables as shown in Figure 2-6.

```
6  int number1
7  int number2;
8  int result;
```

Figure 2-6: Declaring Variables

In every programming language there are strict rules about how everything has to be written. We call this the **syntax** of a programming language. When declaring a variable you need to indicate the **type** of the variable. We will be talking about the possible types of variables in the next section. The variables that we are using in our example program are of type integer. This means that those variables can only store whole numerical values (e.g., 1,34,189, ...). In Java you indicate a variable of type integer by writing 'int' before the name of the variable.

After the name of the variable we have to put a semicolon (;). As you see in the example in Figure 2-3, there is a semicolon (;) at each end of the line. Each of the lines in Figure 2-6 is a statement. The syntax to declare a variable is shown in Figure 2-7.

Syntax to declare a variable in Java:

```
dataType variableName;
```

The following line of code declares a variable of **type** *integer* with the **name** *number1*.

```
int number1;
```

Figure 2-7: Declaring Variables (cont'd)

ASSIGNING VALUES TO VARIABLES

The declaration of a variable is only the first part of using a variable. Once the variable is declared we normally use it to store a value in the variable. To store a value in a variable we use the 'assignment' operator (=), we also say we assign a value to a variable.

Figure 2-8 explains the syntax to assign a value to a variable and gives examples of assigning values to variables of different types.

- To store a value in a variable, use the Assignment operator (=).

- Examples:

```
int counter;  
counter = 7;
```

```
double price3;  
price3 = 34.50;
```

```
String greeting;  
greeting = "Good morning";
```

Figure 2-8: Assigning Values to Variables

VARIABLE NAMES

When choosing a variable name you should not take any name. There are conventions that you should follow when deciding on a variable name. They are listed in Figure 2-9.

- A variable name should describe its purpose (a variable should have only one purpose).
- A variable name must be made up of one or more alphabetical characters, numerical digits, underscore (`_`), or dollar signs (`$`).
- A variable must begin with an alphabetical character, underscore or dollar sign.
- A variable must not be one of the reserved keywords for the language you are using.

Figure 2-9: Variable Names

The list of reserved keywords in Java is provided in the appendix of this book.

These are examples of valid variable names:

`number1`

`result`

`_celsiusValue`

`_fahrenheitValue`

These are invalid variable names:

`3333a` (the variable does not start with an alphabetical character)

`class` (this is a reserved word)

`return` (this is a reserved word)

Exercise 2-1: Find Valid Variable Names



Purpose: Practice identifying valid variable names.

Instructions: In Figure 2-10 valid and invalid variable names are listed. Study the rules for naming variable names that are given above and mark the valid variable names by drawing a circle around them.

\$name	celsius	class	vegetarian
7mark	_counter	mark	\$a
1234	number1	counter	bi

Figure 2-10: “Find Valid Variable Names” Exercise

A list of the valid variable names is provided in Unit02/exercisesol2_1.txt.

What are Data Types?

Every variable in a Java program needs to be of a data type that is defined for the programming language. In our sample program, for example, we are using variables of type integer. There are 5 main data types that are used in Java and in many other programming languages.

The data type determines what sort of value can be stored in a variable. A variable could for example either store a number or a word. It is important to distinguish between different types of data as the program has to handle different data types in different ways.

For example, a program can only calculate with numbers, it cannot make any calculation with words. Words, however, can be printed to the screen. Variables of different types are assigned different amounts of space in the memory of the computer. A variable of type integer for example takes up 32 bits of space, a variable of type double takes up 64 bits of space. A bit is the smallest unit of storage in a computer. Thus, you have to be careful in using the different types; you cannot assign a double value to an integer variable for example as the double value takes more space than the integer variable provides.

The main data types that are used in Java are listed in Figure 2-11.

- **int** - integer, a whole numerical value (e.g. 9)
- **double** - a fractional numerical value (e.g. 56.50)
- **char** - character, an individual letter, digit or symbol (e.g. 'R')
- **boolean** - a boolean variable can have the values true or false
- **String** - words or sequences of letters or numbers

Figure 2-11: Data Types in Java

Now let us look at each of those data types in more detail.

The Data Type integer

Variables of type integer represent whole numerical values. In a program you use a variable of type integer for example to store the value for the number of loops a runner has completed or the number of employees in a company.

In Figure 2-12 the syntax for declaring a variable of type integer and examples are presented. In Java to declare a variable of type integer you write 'int'.

Declaration of **integer** variables:

```
int variableName;
```

Examples:

```
int numberOfLoops;  
int numberOfEmployess;
```

Example values:

4, 56, 70002, 456

Figure 2-12: Variables of Type integer

The example program Unit02/AddingNumbers.java uses variables of type integer. Go back to Figure 2-3 and study lines 6,7 and 8 where the variables of type integer are declared.



Exercise 2-2: Declare a Variable of Type integer

Purpose: Practice declaring a variable using the data type integer by modifying our sample program.

Background: In the program Unit02/AddingNumbers.java we declare three variables (number1, number2 and result). The variable result stores the result of adding the values of number1 and number2. In this exercise we want to sum up three numbers instead of just two numbers. The pseudocode describing this program is as follows:

```
AddingNumbersSol
    Read number1
    Read number2
    Read number3
    result = number1 + number2 + number3
    print result
END
```

Instructions: This is the first program that you are going to edit. We provide you with a template where you are prompted to fill in one line of code: the declaration of a variable. Look for a line of code that starts with `/**`. After this line we want you to fill in the declaration of the variable. In the template we already declared three of the variables for you and we assign values to all of them. We also do the calculation for you and print the result to the screen. So the only thing that is missing in the program is the declaration of the variable number3.

Use the program AddingNumbersTem.java as a starting point and complete the steps as outlined in Figure 2-13.

To complete the exercise, carry out the following steps:

1. Open the file Unit02/AddingNumbersTem.java.
2. Edit the file:
Declare a variable with the name number3 of type integer.
You can use the number2 variable declaration as a model.
3. Save the file.
4. Compile the program.
5. Run the program.

Figure 2-13: “Declare a Variable of Type integer” Exercise

Answer the following questions:

1. What value is assigned to the variable number2? ____
2. Find the line where the result is calculated and copy the whole line into your book.

3. How many variables are being used in this program?

Figure 2-14: “Declare a Variable of Type integer” Exercise (cont’d)

The solution for this exercise is available online in `Unit02/AddingNumbersSol.java`



Exercise 2-3: Assigning Values to Variables of Type integer

Purpose: Practice assigning values to variables of type integer.

Instructions: Follow the steps outlined in Figure 2-15 to complete this exercise.

1. Open Unit02/AddingNumbersA.java.
2. What are the current values?
number1 : _____
number2 : _____
3. Pick new values.
number1: _____
number2 : _____
4. What do you expect the result to be? _____
5. Now change the values of the variables in the Java program.
6. Compile the program.
7. Run the program.
Is the result as you expected?

Figure 2-15: "Assigning Values to Variables of Type integer" Exercise

The Data Type character

Variables of type 'character' represent a single character. The character type encompasses all the letters, digits and symbols that appear on a keyboard.

In a program you might want to use variables of type character for example to store the names of buildings of a university. Think of a university that has buildings that have single characters as the identifying name: The Math building 'A', the Physics building 'B', the Music department building 'C'.

Variables of type character are also useful for the options in a menu system for a program. A computer program might offer the options 'p' for printing, 's' for saving and 'e' for exiting a word processing program for example. The options are represented as character variables in the program.

In Figure 2-27 the syntax for declaring a variable of type character is shown. In Java to declare a variable of type character you write 'char'.

The syntax for declaring a variable of type **character** is as follows:

```
char variableName;
```

Examples:

```
char mathBuilding;  
char physicsBuilding;
```

Example values are:

'a', 'C', '@'

Figure 2-27: Declaring a Variable of Type character

The program Unit02/FindBuilding.java as shown in Figure 2-28 uses variables of type character. The pseudocode for this program is shown below:

```
FindBuilding  
    read mathBuilding  
    read physicsBuilding  
    print "The Math department is in building mathBuilding"  
    print "The Physics department is in building  
                                           physicsBuilding"  
END
```

```
1  class FindBuilding {
2
3  public static void main(String[] args) {
4
5      // Declare two variables of type character
6      char mathBuilding;
7      char physicsBuilding;
8
9      // Assign values to the variables
10     mathBuilding = 'A';
11     physicsBuilding = 'B';
12
13     System.out.println("The Math department is in building " +
14                        mathBuilding);
15     System.out.println("The Physics department is in building "
16                        + physicsBuilding);
17 }
18 }
```

Figure 2-28: FindBuilding.java

In lines 6 and 7 the variables of type character are declared. Then in lines 10 and 11 we assign values to the variables. Note that values of type character always need to be inside single quotes (''). That is what the syntax of Java requires.

Compile and run the program as shown below.



```
C:\o1303\UNIT02>javac FindBuilding.java
C:\o1303\UNIT02>java FindBuilding
The Maths department is in building A
The Physics department is in building B
C:\o1303\UNIT02>
```



Exercise 2-8: Declare a Variable of Type character

Purpose: Practice declaring a variable using the data type character.

Background: In the program Unit02/FindBuilding.java we declared two variables of type character, one for the Math building and one for the Physics building. In this exercise we want you to declare a third variable of type character to identify the Music department of the university.

The pseudocode for this program is as follows:

```
FindBuildingSol
    Read mathBuilding
    Read physicsBuilding
    Read musicBuilding
    print the values of the three variables
END
```

Instructions: We provide you with a template. You are prompted to fill in one line of code: the declaration of the variable. The rest of the code is already written for you.

Use the program Unit02/FindBuildingTem.java as a starting point and complete the steps as outlined in Figure 2-29

To complete the exercise, carry out the following steps:

1. Open the file /Unit02/FindBuildingTem.java.
2. Edit the file:
Declare a variable of type char with the name musicBuilding.
3. Save the file.
4. Compile the program.
5. Run the program.

Figure 2-29: “Declare a Variable of Type character” Exercise

- Try to come up with other ideas of other situations where you would use the data type character.
- Where could this data type be useful?

Figure 2-30: “Declare a Variable of Type character” Exercise

A solution can be found in Unit02/FindBuildingSol.java.

Unit Review Questions

Given the program below answer the following questions:

```
1  class SummingPrices {
2
3  public static void main(String[] args) {
4
5      double price1;
6      double price2;
7      double result;
8
9
10     price1 = 234.55;
11     price2 = 40.00;
12
13     result = price1 + price2;
14
15     // Print out the sum
16     System.out.println("The result of adding price1 and price2 is "
17 +
18     result);
17     }
18 }
```

1. What variables are declared. List them:

2. What are their types?

3. Which operator is being used in line 13?

4. What will be the value of the variable sum?

5. In which line does the main method start?

6. In which line does the main method end?
7. What other operators could you use in line 13?
8. Why can't you use the concatenation operator in line 13 without changing the type of the variables?
9. What would be the value of the variable result if you were using the subtraction (-) operator with the given values?
10. Which sign is always at the end of a statement?
11. Is number1 a valid variable name?
12. Is 7math a valid variable name?
13. List three integer values.
14. Could you assign an integer value (e.g. 15) to a double variable without getting an error when you compile the Java program?
15. What are the two possible boolean values?
16. List three values of type character.
17. What does method println do?
18. Which are the operators that can be used for integer as well as double variables?
19. What is the result of the following expression? Why?
 $8 + 4 * 3$
20. What are the two types of comments in Java?