

MT201

Unit 1

Computer concepts and first Java program

Course team

Developer: Herbert Shiu, Consultant

Designer: Dr Rex G Sharman, OUHK

Coordinator: Kelvin Lee, OUHK

Member: Dr Reggie Kwan, OUHK

External Course Assessor

Professor Jimmy Lee, Chinese University of Hong Kong

Production

ETPU Publishing Team

Copyright © The Open University of Hong Kong, 2003.
Reprinted 2005.

All rights reserved.

No part of this material may be reproduced in any form
by any means without permission in writing from the
President, The Open University of Hong Kong.

The Open University of Hong Kong
30 Good Shepherd Street
Ho Man Tin, Kowloon
Hong Kong

Contents

Course overview	1
Introduction	3
Objectives	4
Problems and computer solutions	5
Real-world problems	5
Manual solutions versus computer solutions	5
Problems that are difficult to solve using current technology	9
Basic computer structure	12
The history of computers	12
Hardware	16
Software	29
Programming languages	30
How programs are executed	33
The Internet and the World Wide Web	36
The Internet	36
The World Wide Web	38
Java, what is it?	40
Why object-oriented programming and Java were chosen for <i>MT201</i>	40
Java applications and applets	44
Installation of Java Software Development Kit	46
Compiling and executing Java programs	57
Executing a Java demonstration program	57
Your first Java program	57
Summary	63
References	64
Glossary	65
Suggested answers to self-test questions	70
Feedback on activities	72

Course overview

Welcome to *MT201 Computing Fundamentals with Java*.

MT201 is one of the core courses in the Computing and Networking, Applied Computing, and the Electronic Commerce programmes offered by the OUHK. It may be your first course in computer programming. Your OUHK course development team is well aware of this, so the team has made the conscious decision to develop this course on the basis that you have no prior knowledge or experience of programming in any programming language. Therefore, the material is presented step-by-step and involves many examples and programming exercises. We believe the only way to become an effective programmer is for you to develop your own programs and test them.

By basing this ‘first’ course on the Java programming language, you are actually being positioned at the cutting edge of programming. How can we make such a claim, you might ask? If a programming language other than Java had been chosen to support your learning of computer fundamentals, you might easily have missed out on learning about objects. Over the last ten years or so, programming has moved from imperative programming, in which program codes were difficult to reuse, to object-oriented programming, in which program codes can be used over and over again. Although there is a lot to be gained from learning the logic of programming through step-by-step (imperative) programming, the reality is that object-oriented programming is a more natural methodology for solving problems. And the working objects you develop essentially become your own ‘library’ of objects — you can take them out again and again and have them work for you.

Perhaps the key idea to be aware of as you start this course is that there is no ‘one correct way’ to program. Some ways are better than others; what you are really trying to create are efficient and, of course, effective programs. Programs must be effective to achieve their goals (who would try to develop an ineffective program anyway?) and so that they conserve resources like running time, memory and even maintenance time. Having said that there’s no one correct way to program, we must alert you right up front that any programming language (Java included) has many rules that must be followed and programming conventions (practices) that should be followed. After you learn the rules and conventions, you’ll be well on the way to becoming an efficient and effective programmer.

The textbook, *Java Programming: From the Beginning* by King (we usually refer to this as ‘your textbook’ or simply as ‘in King’), was designed as a primary text for first-year college or university courses in the US. According to the author, ‘no previous programming experience is necessary’ (p. xxi). Each chapter contains:

- style and design tips
- warnings to alert you to common programming pitfalls (traps)
- numerous review questions

- a Q & A section
- exercises
- answers to the review questions and, in later chapters,
- program segments to illustrate effective programming in Java.

We would also like to draw your attention to *Java Language Summary* in Appendix B, beginning on page 665 in King.

Although you will be directed to read specific parts of the textbook as you work your way through the units, as a university student you shouldn't wait to be asked. If a term appears in the unit that you don't understand clearly (having read the explanation in the unit), look it up in the index and read about it in the textbook. Also, the text provides another way of explaining the information covered in the units, so you can always get a second description even if you understood what was explained in the unit. By the way, as you read this unit you'll find certain words printed in **bold**; the meanings of those words are defined in the 'Glossary' at the end of the unit.

We would also strongly encourage you to attempt as many of the review questions and exercises as you possibly can — the more you challenge yourself, the better and broader your learning will be. Self-tests and activities will be included in each study unit, but you should regard these as the basic set of ways in which you can test your understanding. Extend yourself, and use your textbook to the limit (after all, you paid for it) to get the most out of *MT201* that you possibly can.

If you do encounter problems that you really can't understand (having used the study unit and the textbook), contact your tutor *immediately*. Don't delay in seeking help — if you straighten out a problem when you first identify it as a problem, you'll be able to understand the work that comes later. It's rather like getting back on the right path if you happened to start out wrongly. If you delay, however, getting your learning back on the right path becomes harder. Your tutor is there to help you to learn and to avoid frustrations in your learning.

Finally, programming is fun — even though it can be challenging at times. Experienced programmers feel a great sense of satisfaction once their programs do exactly what they want them to do. *Learning* to program is challenging too, but it can also be fun. Keep your tasks simple at first, learn the concepts one by one, learn and apply the rules and conventions and then — *be creative*. There's no one correct way to program *solutions*.

Enjoy your *MT201* experience.

Introduction

MT201 Computing Fundamentals with Java is an elementary course for computer-related disciplines in the Open University of Hong Kong. This course covers some elementary knowledge in computer science and teaches you how to write programs in an object-oriented programming language, Java.

Unit 1 discusses what computers are and what makes them so important in the modern world. Even though computers are operating at faster speeds, there are still problems that cannot be solved by computers. Therefore, we discuss the nature of problems that can be handled and solved by computers.

In order to enable a computer — an electronic device — to perform in a predetermined way, a programmer must provide it with a sequence of instructions so that it can operate accordingly. Such a sequence of instructions is known as a program or **software**. Each instruction is written in a **programming language**. *Unit 1* discusses a modern software development approach — the object-oriented approach and the **object-oriented programming** language — and why such an approach is preferable to other approaches.

Many object-oriented programming languages are available nowadays, but Java has become extremely popular since its introduction in 1995. *Unit 1* discusses why the Java programming language was chosen for this introductory course. Furthermore, we provide you with the details for setting up your computer so that you can start writing Java programs using your own computer.

Objectives

On completing *Unit 1*, you should be able to:

- 1 *Identify* the problems that can be solved using computers.
- 2 *Describe* the basic components of a computer.
- 3 *Describe* the relationship between hardware and software.
- 4 *Distinguish between* interpretation and compilation.
- 5 *Describe* and *use* the Internet and the World Wide Web.
- 6 *Describe* the Java program development and execution model.
- 7 *Install* and *use* the Java Software Development Kit.

Problems and computer solutions

Every day, you are likely to encounter many problems that you have to handle — from preparing breakfast in your home to preparing management reports in the office. It doesn't matter whether the problems are simple or complicated, *ad hoc* or recurring; they need to be handled properly. In the past, you might have solved them manually, and that probably took a lot of time. Now, you can handle some of them easily through the use of computers.

The following section discusses various types of problem and the way computers handle them.

Real-world problems

Imagine that you were asked to write down all the problems you encounter in a day. For example, some of the problems might include:

- 1 preparing a meal for your family
- 2 preparing a letter
- 3 calculating the monthly income and expenditure balance sheet
- 4 maintaining a pile of name cards.

Manual solutions versus computer solutions

In the past, when electronic tools didn't exist, the above-mentioned problems still had to be solved. For example, you used a gas cooker to prepare a meal, a typewriter (probably with a partially worn-out ribbon) to type a letter, an abacus to prepare the balance sheet and a name-card holder to store all the name cards you had collected.

If all these problems could be solved without computers in the past, why is the computer so 'irreplaceable' nowadays? First of all, let's see how you can solve the above problems with computers.

- 1 You can use a microwave oven to prepare a meal. By using the default cooking programs provided, you can prepare different styles of food. You can even set a timer so that the oven can start at the pre-set time. A microwave oven can also be considered a computer. We discuss this in the following sections.
- 2 You can prepare the letter using a word-processor, which can also help you proofread your document by checking on things such as spelling and even grammar. You can print the document with a laser printer to get a publisher-quality printout.
- 3 A balance sheet can be easily prepared using a spreadsheet. What's more, you can visualize your financial status by generating a pie chart diagram.

- 4 Using a name card scanner with proper supporting software, you may scan all of your name cards so that all **data** on the name cards such as name, job title, phone number, facsimile number, email address, address and so on are extracted and stored in a database. The database can then be transferred to your handheld Personal Digital Assistant (PDA) that you can carry around with you.

Compared with the manual solutions, can you work out the benefits that computers bring us?

- 1 *Accuracy.* A computer is basically a machine that can perform calculations. In preparing the monthly balance sheet, using a spreadsheet application can minimize human intervention. You just need to ensure the raw data items (and formulas) in the balance sheet are correct, and then the final balance will be correct. Another example is using an electronic dictionary to check the spelling in a document. You can be sure that the checking is more accurate than scanning by eye.
- 2 *Speed.* To compare the speeds of using different solutions, we can compare the times required to complete the same task. The shorter the time to complete a task, the greater the speed of the solution. For example, you can change an item in the spreadsheet and the balance is updated instantly. Otherwise, you have to trace and amend the items one by one.
- 3 *Availability and convergence.* Advances in computer technology also bring availability (ease of access). You can now use a handheld PDA to maintain all your schedules, address book and memos so that you no longer need to carry a bulky diary and a pile of name cards along with you. If your handheld PDA can access the **Internet** via a wireless network, you can even access the latest information in your office when you are on the road.
- 4 *Automation.* Like using a microwave oven, you can pre-set the time and program the functions so that the oven will perform according to the settings. Similarly, you can set a timer in your handheld PDA to remind you of important appointments and events.

Data and algorithms

Data

No matter what task you want to complete, you need some basic information or raw materials. For example, in preparing the balance sheet, the items (the ‘numbers’) are the raw materials that you want to manipulate. In cooking with a microwave oven, the length of time required and the food are the basic information and the raw materials respectively. From the perspective of a computer, these raw materials are known as data.

There are two categories of data — analogue and digital. Data represented using physical magnitudes such as temperature, electric

current or voltage, and frequency are analogue and they are real numbers. Because most real numbers cannot be represented exactly (and cannot be read off from any physical device exactly either), analogue data are stored in computers as approximations. However, digital data can be represented exactly in computers.

In general, digital computers (the type of computer widely used nowadays) can only manipulate digital data. Analogue data must be converted to an equivalent digital format so that the computer can process them. The conversion device is known as an **analogue-digital converter** (AD converter). The results of computer processing are also in digital format. Sometimes, if you need to reproduce the data in analogue format, a **digital-analogue converter** (DA converter) will be needed for the conversion.

Information and data are two similar terms. For most uses, they are interchangeable. Data, strictly speaking, are items before processing. The meaningful result obtained after processing is known as information.

Algorithms

In order to obtain the necessary result, a well-defined formula or sequence of steps is applied to the data. Such a formula or sequence of steps is known as an **algorithm**. For example, you need a set of rules to use the abacus for calculation. Such a set of rules is an algorithm. Another real-life example is finding a word in an English dictionary. All words in a dictionary are sorted from the letters A to Z. The first word and the last word on each page are shown in the page header. If you want to find the explanation of a word, say 'computer', you locate the section of the letter 'C' first. Then, you search the headers of the pages in the section sequentially to find out whether the word 'computer' is either a word in the header or between the two words shown in the header. When the page is found, you go through the page to find the word. It is either on the page with an explanation or, if it isn't there, it is not included in the dictionary. You can see that you are actually using some kinds of algorithm every day, but you may not be aware of it.

Algorithms have to be presented in a precise way for readers to follow, in order to avoid ambiguity. For example, the above algorithm for searching a word in a dictionary can be presented as:

- 1 Locate the first page of the section that is the same as the first letter of the word you are looking for.
- 2 For each page in the section, search for the word by checking the range of words in the header:
 - a If the word you are looking for is out of the range, go to the next page and repeat step 2.
 - b If the word you are looking for is shown in the header of the page, the page for the word is found.

- c If the word falls within the range of the words in the header, scan through the page to determine whether the word can be found in the dictionary.

Reading

King, Section 1.3, pp. 10–13, gives a brief introduction to programming and algorithms. Then attempt the review questions on p. 13. You can find the answers on p. 26.

Automatons

People are reluctant to perform tedious and repetitive tasks. Therefore, human beings have been inventing various machines that will automate those tasks. The machines that can follow a predetermined sequence of steps are known as *automatons*.

An important historical automaton was the Jacquard loom invented by a French weaver, Joseph-Marie Jacquard, which used punch cards to control the patterns to be woven. The weaving machine followed the punched holes on the cards and produced the desired patterns. It was revolutionary because it enabled information (the weaving patterns) to be stored on cards, and the punched cards could be used over and over again.

Punched cards were used in the US census of 1890 and were a great success. For the 1880 census, the amount of data collected was huge and took almost ten years to process, by which it was time to conduct the next census — 1890! For the 1890 census, an army engineer, Herman Hollerith, invented Herman Hollerith's Tabulating Machine that used a punched card to store the information from each person. The machine processed the punched cards by pushing pins against them. If there was a hole at a particular position of the card, an electric circuit was completed and the count of the corresponding item was increased by one. Otherwise, the count was unchanged. The machine helped the US Government to complete the processing in less than three years, a great improvement over the 1880 census. Because of the success of the machine, Hollerith formed a company that later became the International Business Machine Company, or IBM.

The use of punched cards not only occurred in the weaving industry and US censuses but also in computers. Punched cards (and its variant punched tapes) were used with early computers until the 1970s. They could be used to store programs as well as data. One punch card could store 80 (columns of) characters, which corresponds to the 80-column line of a text editor used today. One advantage of punched cards was that they could be prepared with a mechanical keypunch machine without connecting to a computer. This saved expensive processing power of the computer at that time. After the cards were prepared, a punched card reader could read the data stored in the cards into a computer.

Problems that are difficult to solve using current technology

Can computers solve all problems within a reasonable time? (Depending on the application involved, a reasonable time may range from seconds to years.)

At the moment, the answer is definitely negative. A computer can only solve a problem within reasonable time if there is an efficient algorithm and the data set given is not too large.

A computer can be treated as a programmable calculator. You have to provide the instructions for it to follow, or it can do nothing. For example, look at the following quadratic equation:

$$x^2 + 3x + 2 = 0$$

You want to find its roots (the values of x that make the right-hand side 0). According to the findings of some mathematician or other, we know that for a quadratic equation in the following format,

$$ax^2 + bx + c = 0$$

the possible values of x are given by

$$x = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad (\text{Solution 1})$$

and

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad (\text{Solution 2})$$

Therefore, you could ‘program’ the formulas into your computer. It could then follow your instructions and obtain the solutions (the possible values in D2 and E2 below) for you. You could use a desktop calculator to get the values of x by substituting the values of a , b and c in the above formulas. If you want to use a computer to determine them, you can, for example, ‘program’ the above formulas in a spreadsheet, like in Figure 1.1:

	D2		=	=(-B2-SQRT(B2*B2 - 4*A2*C2))/2/A2		
	A	B	C	D	E	F
1	a	b	c	Solution 1	Solution 2	
2	1	3	2	-2	-1	
3						

Figure 1.1 Calculating a quadratic equation solution using a spreadsheet

Software is available that gives you the desired solutions immediately after you provide the values a , b and c . It seems that you have not presented or programmed any algorithm to it, but actually the developers

of that software have already presented the formula (or the algorithm) to the computer in another way — you are simply not aware of it. The key point is that computers need the related algorithm(s) to solve the problems.

Therefore, a computer is simply a machine for performing calculations. It helps you perform complicated calculations, but the discovery of the underlying theories and formulas are within the scope of other scientific areas.

Currently, there are still many problems for which researchers cannot derive efficient algorithms to solve, and they simply try all possibilities to see whether any one (or some) of them is the solution — the so-called trial-and-error approach. A typical example is the travelling salesperson problem.

The idea of the travelling salesperson problem is that a salesperson has to travel to n different locations exactly once and return to the start location afterwards. The travelling times from one location to another are different. From one location, the salesperson can travel to all other locations directly. The problem is to determine the minimum travelling time and the most desirable path to the locations.

Researchers believe that there is no efficient algorithm for the problem; it is only possible to try all possible paths and choose the one with the minimum travelling time among all. You might think that computers are so fast that it is possible to try all possibilities in a short time. Let's see how long a computer needs to solve the travelling salesperson problem if there are n locations to be visited and the computer can try 1,000,000 possibilities in a second (which is quite slow).

The number of possible paths is determined by considering that, at the start location, the salesperson can choose one of the n locations as the first location to visit. The salesperson travels to the first location and then can choose one of remaining $(n-1)$ locations to be the next, and similarly for the rest of the locations. Therefore, the total number of possible paths is given by

$$n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1$$

which is called the n -factorial, denoted by $n!$. The following table shows the times required for different values of n :

The value of n	Number of possible paths ($n!$)	Time required
10	3628800	3.6 seconds
15	1.30767×10^{12}	15.1 days
20	2.4329×10^{18}	77147 years
100	9.3326×10^{157}	2.9594×10^{144} years

You can see that even though we have a systematic way to solve the travelling salesperson problem — and it is possible to present such an approach to the computer — the time required to solve it increases exponentially with the number of locations in the problem. For a value of 20, which doesn't seem to be a big value, it is already impossible to complete the whole process during a person's lifetime.

Self-test 1.1

- 1 Please suggest an algorithm, the sequence of steps you might perform, when you check out at the supermarket cashier.
- 2 ZIP is a *de facto* file format for storing one or more files with compression for easy distribution across the Internet and for saving storage space. Furthermore, it enables password protection so that the stored files can only be extracted by presenting the same password as when the ZIP file is created.

If you created a ZIP file but forgot the password, you could find a software application to recover the password by trying all possibilities. If you have a computer that can try 1,000,000 passwords in a second, determine the times required for recovering the passwords in the following formats:

- a a word that can be found in a dictionary (assume that an English dictionary contains 200 million words),
- b an arbitrary combination of 8 lowercase letters,
- c an arbitrary combination of 8 lowercase or uppercase letters, and
- d an arbitrary combination of 8 lowercase letters, uppercase letters and digits (i.e. 0,1,...,9).

Basic computer structure

You occasionally hear on the news that new computer technologies — such as new computers, new processors, new **operating systems** and even viruses — have been invented. But what are they? Are new computers significantly different from the old ones?

Since computers first appeared, their structure has not changed much — new computers are roughly the same as the old ones. In the following section, we first discuss the history of computers and then computer structures.

The history of computers

A computer is a device that performs calculations based on given data and pre-defined steps. Therefore, a desktop calculator can also be considered a computer.

Throughout history, various devices have been invented to help people perform calculations. Such devices might have been mechanical or electrical; they range from large devices that occupied 1,800 square feet in the past to the mini size that we can now place in our palm.

Calculating devices

People tend to be facing increasingly complicated problems. An example is that the analysis of a population census involves more and more calculations, as the population is getting bigger. As a result, people have invented many devices that were dedicated to performing mathematical operations. Before the invention of electronic calculation devices, those calculation devices were mostly mechanically driven.

A few important calculating devices are discussed below.

Abacus

The abacus has been around since 1000 BCE and is considered to be the earliest automaton. The modern abacus, which appeared in China in the 13th century, is made up of 13 rods — each with two beads on top and five beads below (see Figure 1.2).

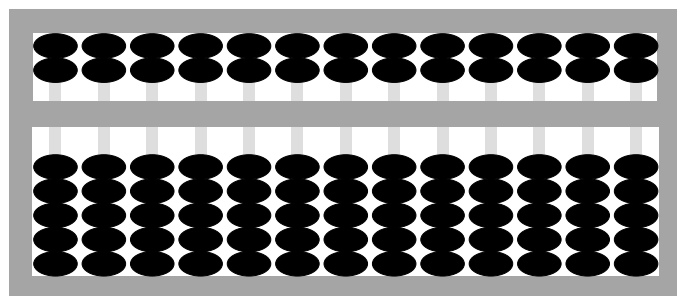


Figure 1.2 An abacus

In the 17th century, the Japanese borrowed the idea from the Chinese abacus and invented the Japanese-style abacus that has 21 rods with one bead on top and four beads below. If you want to know more about abacus, you can visit

<http://www.ee.ryerson.ca:8080/~elf/abacus/intro.html>, which contains explanations and a photograph.

Napier Bones

In the Western world, John Napier invented a device in 1612 (shown in Figure 1.3) known as the Napier Bones. It was dedicated to performing multiplications — with the limitation that one of the numbers must be a single digit. The idea underlying such a device was that it had rods (columns) that corresponded to a number (consecutive numbers from 1 to 9) in its multiplication table of products. The rods were:

	0	1	2	3	4	5	6	7	8	9
1	0 0	0 1	0 2	0 3	0 4	0 5	0 6	0 7	0 8	0 9
2	0 0	0 2	0 4	0 6	0 8	1 0	1 2	1 4	1 6	1 8
3	0 0	0 3	0 6	0 9	1 2	1 5	1 8	2 1	2 4	2 7
4	0 0	0 4	0 8	1 2	1 6	2 0	2 4	2 8	3 2	3 6
5	0 0	0 5	1 0	1 5	2 0	2 5	3 0	3 5	4 0	4 5
6	0 0	0 6	1 2	1 8	2 4	3 0	3 6	4 2	5 8	5 4
7	0 0	0 7	1 4	2 1	2 8	3 5	4 2	4 9	5 6	6 3
8	0 0	0 8	1 6	2 4	3 2	4 0	4 8	5 6	6 4	7 2
9	0 0	0 9	1 8	2 7	3 6	4 5	5 4	6 3	7 2	8 1

Figure 1.3 Napier bones for multiplying numbers involving at least one single digit number

To calculate the product of 2468 and 7, first place the rods for 2, 4, 6 and 8 side-by-side sequentially and identify the row for digit 7. That is:

	2	4	6	8
1	0/2	0/4	0/6	0/8
2	0/4	0/8	1/2	1/6
3	0/6	1/2	1/8	2/4
4	0/8	1/6	2/4	3/2
5	1/0	2/0	3/0	4/0
6	1/2	2/4	3/6	5/8
7	1/4	2/8	4/2	5/6
8	1/6	3/2	4/8	6/4
9	1/8	3/6	5/4	7/2

Figure 1.4 An example to find the product of 2468 and 7

Then the following region gives the result of the multiplication by getting the sums in the parallelograms:

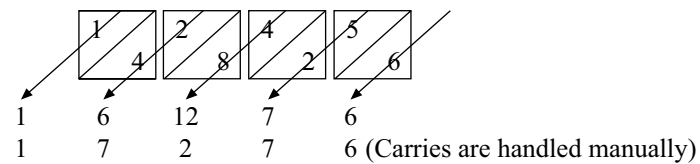


Figure 1.5 Finding the sums and handling carries

Therefore, the product of 2468 and 7 is determined to be 17276. You can see that the device can help in performing multiplications without memorizing the multiplication tables.

Slide rule

John Napier not only invented the Napier Bones that help in multiplication operations, he also invented the logarithm. Based on the theory of logarithms, William Oughtred created the slide rule. Engineers used it until the early 20th century, even though its accuracy was only up to three digits.

Pascalene

A mathematician, Blaise Pascal, invented a mechanical adding machine in 1642, named the Pascalene. It contained dials and gears so that if the sum of two digits was greater than or equal to 10, the carry was automatically brought to the next position.

Programmable computers

Computers are powerful mainly because they can accept sequences of instructions and apply the operations accordingly to the data to get a result. Such devices that can accept sequences of instructions — programs — and perform them are known as programmable computers.

Early computers

The first programmable computer was a steam-powered computer named the Analytical Engine, invented by Charles Babbage in 1833. It was a mechanical digital computer that used punch cards to encode the instruction to be executed. Although the engine was never completed, it did contain the components of modern computers, so Charles Babbage is therefore known as the 'Father of the computer'.

About 100 years after Charles Babbage, a group of researchers at Harvard University (sponsored by the IBM Company) built the Mark I computer in 1937. It used thousands of mechanical binary switches that were electrically controlled. A major difference from modern computers, though, was that it stored and manipulated decimal numbers. About the same time, a German engineer, Konrad Zuse, developed a similar programmable electronic digital computer in 1941.

The first generation of modern computers

The significance of the first generation of modern computers was that they were built with electronic **vacuum tubes**, which functioned as controllable on-off switches. The vacuum tubes were not reliable, however, and they consumed a lot of electrical power.

In 1945, the first successful general-purpose electronic digital computer known as the **Electrical Numerical Integrator and Calculator (ENIAC)** was developed at the University of Pennsylvania. It was built with 18,000 vacuum tubes and could perform 300 multiplication operations per second. This was really a giant machine since it occupied 1,800 square feet of floor space.

Based on the experiences of developing the ENIAC, a consultant on the ENIAC project, John von Neumann, proposed a number of improvements over the ENIAC design. The model he proposed is known as the **von Neumann architecture** (discussed later).

Within six years, the Electronic Discrete Variable Automatic Computer (EDVAC) and the first commercially available computer, the Universal Automatic Computer (UNIVAC), were invented based on the architecture proposed by von Neumann.

The second generation

The invention of **transistors**, which are made from semi-conducting material, brought about the second generation of computers that lasted from 1956 to 1963. Semi-conducting means the material could conduct (on) or not conduct (off) an electric current. With transistors as the new on-off switching devices, the computers at that time worked faster,

consumed less electrical power, and became more reliable and less bulky. One example of these second-generation computers was the Livermore Atomic Research Computer (LARC), which could even fulfil the needs of atomic scientists but was too costly for the business sector. At that time, computer companies released some other versions of computers that businesses, the government and universities could afford.

The third generation and the future

In 1958, the **Integrated Circuit (IC)** (also known as a chip) was introduced. It integrated a lot of transistors (and other components) to form an electric circuit. As it was smaller and consumed even less power (and hence less heat was generated), it quickly replaced transistors in building computers and was widely used from 1964 to 1971.

The advance in technology brought about Large Scale Integration (LSI), Very Large Scale Integration (VLSI) and Ultra-Large Scale Integration (ULSI), which increased the number of components that could be stored on a chip. Computers could therefore be built much smaller — and at a much lower price.

In 1971, Intel developed a general-purpose processor with the code number 4004 that could be programmed for use in different devices. Such processors were used in household electrical appliances such as microwave ovens.

In response to the demand from the general public, computer companies entered the consumer market by designing products that met the needs of the consumers. Successful products included the Commodore and Apple computers.

IBM introduced the **personal computer (PC)** in 1981. Three years later in 1984, Apple introduced the Macintosh series of computers, which enabled users to perform operations by using a ‘mouse’ in addition to the keyboard.

Computers have become smaller in physical size, larger in **memory** size and worked faster since the first generation. According to **Moore’s Law** established in 1965 by Gordon Moore, the co-founder of Intel, computer processing power will double every 18 months. This is expected to hold true throughout the next 20 years.

Hardware

The components that you can physically touch in a computer are the *hardware* of the computer.

Computers can be categorized into general-purpose and special-purpose. Our first impression of a computer is the desktop kind, with a monitor, a keyboard and a mouse. Such computers are known as **general-purpose computers**, as they can be used for multiple purposes. If a computer is designed for a specific purpose, like a TV game machine or a digital

camera, it is a **special-purpose computer**. A mobile phone is also a special-purpose computer, as it contains the basic components of a computer but its sole purpose is for communication.

According to the physical size and the computation power of computers (that is, the number of operations that can be performed in a second), general-purpose computers can be classified in the following categories:

- 1 *Personal computers*. These are desktop computers that are equipped with a monitor, a keyboard and a mouse. They are traditionally called microcomputers.
- 2 *Workstation*. This is a computer with better computational power than that of personal computer. Some workstations are designed for graphic design. Such workstations are usually equipped with a higher quality monitor and a faster display module.
- 3 *Mini-computers*. These can support more than one user's operations at the same time. They usually use more than one processor, and their computational power is greater than that of a personal computer.
- 4 *Mainframe computers*. These are even more powerful multi-user computers than mini-computers. For an organization or an enterprise, there may be one or two mainframe computers that provide core computational facilities to all users.
- 5 *Supercomputers*. These are characterized by their extremely high processing speeds; that is, they can perform billions of computational instructions per second. Such computers are usually used for scientific and military research, weather prediction and other simulations.

Before we look at other components of a computer, you should first understand the way computers handle data — more particularly, at the way computers represent numeric values.

Computers are not human. They do not use ten fingers for counting and they therefore would not use ten digits to represent a quantity. There are only two states in computers — on and off. As a natural consequence, computers natively use the **binary number system**, a number system with 0 and 1 digits only. For example, the quantity 87 is equivalent to the binary number 01010111. If one component in a computer transfers the quantity 87 to another component, the following electric signal can be transferred through a series of eight cables, as shown in Figure 1.6:

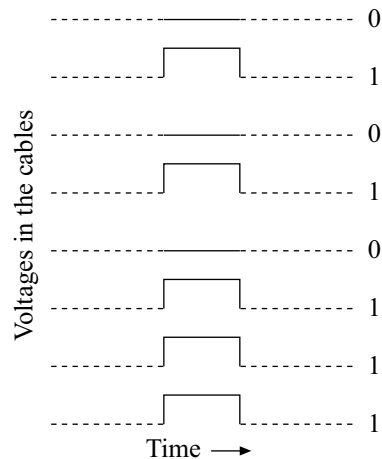


Figure 1.6 Transmitting a binary number 01010111

The above figure shows the voltages in the eight cables at a particular time. The high voltage and low voltage are considered to be 1 and 0 respectively. The receiving component recovers the number by detecting the voltage in the cables. By using such a method, the number 87 is transferred through the electrical cables using the binary number system.

To represent alphanumeric data such as the letter 'A', binary numbers are also used. The letter 'A' is represented by the binary number 01000001 in ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange). This code is used in our PC-compatible machines and many other computers.

Almost all storage technologies use the two-state physical property of a medium for storing the data on them, such as the magnetic polarities on a tiny portion of a floppy diskette. You can see that the binary number system is closely related to the computer design; it is therefore important that you understand such a numbering system.

Following the discussions on numbering systems, we discuss the famous von Neumann architecture, the blueprint of all modern computers.

Number systems

We are familiar with using ten digits, 0 to 9, for representing numeric values, known as the decimal number system. However, computers can only handle two states — on or off, that is, either 0 or 1 — as in the binary number system.

In the decimal number system, the actual quantity represented by the number 1234 is:

$$\begin{aligned} 1234_{10} &= 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 \\ &= 1000 + 200 + 30 + 4 \\ &= 1234 \end{aligned}$$

We use the subscript 10 to denote that the number system being used with 1234 is 10. We can use a similar method to determine the quantity represented by the following binary number:

$$\begin{aligned}
 1011_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
 &= 8 + 0 + 2 + 1 \\
 &= 11
 \end{aligned}$$

Therefore, the number 1011_2 is equivalent to 11 in the decimal number system (11_{10}).

A string of 0's and 1's is difficult for us to read, let alone understand. Therefore, we prefer to handle the numbers in a shorter format; the baseline requirement is that it can be easily converted to and from the binary number system. As a compromise of readability and ease of manipulation by a computer, we prefer numbers to be presented in the octal (base 8) and hexadecimal (base 16) number systems. They are internally stored using the binary number system in computers.

The **octal number system** uses eight digits, 0 to 7 for representing a number (just as we use 10 digits in decimal number system). Let's consider the octal number, 456_8 :

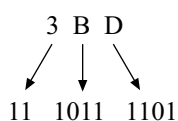
$$\begin{aligned}
 456_8 &= 4 \times 8^2 + 5 \times 8^1 + 6 \times 8^0 \\
 &= 256 + 40 + 6 \\
 &= 302_{10}
 \end{aligned}$$

Similarly, there are 16 digits in the **hexadecimal number system**. They are 0 to 9 and A to F. 'A' represents the value 10, 'B' represents the value 11 and so on. The last one is 'F', which represents the value 15. For example, the hexadecimal number $3BD_{16}$ represents:

$$\begin{aligned}
 3BD_{16} &= 3 \times 16^2 + 11 \times 16^1 + 13 \times 16^0 \\
 &= 768 + 176 + 13 \\
 &= 957_{10}
 \end{aligned}$$

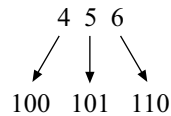
To represent a value of 957 in the binary number system, the number is 1110111101_2 . It is generally easier for us to handle a shorter string of numbers ($3BD_{16}$) in hexadecimal digits.

As mentioned above, the advantage of choosing octal and hexadecimal numbers is that it is much easier to convert octal or hexadecimal numbers to binary numbers than to convert decimal numbers. We can easily convert the hexadecimal number $3BD_{16}$, into the equivalent binary number by converting the hexadecimal digits to the corresponding binary digits individually and combining together. The binary numbers for 3_{16} , B_{16} (11_{10}) and D_{16} (13_{10}) are 11_2 , 1011_2 and 1101_2 respectively. Then, $3BD_{16}$ is equivalent to 1110111101_2 as a binary expression and the process can be visualized as:



Similarly, the process of converting an octal number to the corresponding binary number is to convert each octal digit into the corresponding binary

numbers and then combine them. For example, the binary numbers for 4_8 , 5_8 and 6_8 are 100_2 , 101_2 and 110_2 respectively. Therefore,



the number 456_8 is equivalent to 100101110_2 in the binary number system.

The processes to convert binary numbers into octal numbers and hexadecimal numbers are just the reverse of the above processes. Given a binary number, simply group the binary digits into groups of three digits (for getting octal numbers) or four digits (for getting hexadecimal numbers) from right to left and then convert each group of digits into the corresponding digit in those number systems. For example, the ways to get the corresponding octal number and hexadecimal numbers for a binary number 11000101 are:

$$11000101 = 11\ 000\ 101 = 305_8$$

$$11000101 = 1100\ 0101 = C5_{16}$$

You can see that the processes of converting between binary numbers and octal or hexadecimal numbers are trivial and easy; that is the reason the numbers presented to the users are usually in octal or hexadecimal numbers, whereas they are stored internally as binary numbers. This topic of number systems is covered extensively in another OUHK course *MT260 Computer Architecture and Operating Systems*.

Self-test 1.2

- 1 Convert the following numbers into decimal numbers:
 - a 10101101_2
 - b 6457_8
 - c $ABCD_{16}$
- 2 Convert the following octal and hexadecimal numbers into binary numbers:
 - a 23_8
 - b $A1B2_{16}$
- 3 Convert the following binary numbers into the equivalent octal numbers and hexadecimal numbers:
 - a 11110000
 - b 100010001

Activity 1.1

The following is a game for guessing the age of a person. First, please look at the following table:

1	2	3	4	5	6
32 33	16 17	8 9	4 5	2 3	1 3
34 35	18 19	10 11	6 7	6 7	5 7
36 37	20 21	12 13	12 13	10 11	9 11
38 39	22 23	14 15	14 15	14 15	13 15
40 41	24 25	24 25	20 21	18 19	17 19
42 43	26 27	26 27	22 23	22 23	21 23
44 45	28 29	28 29	28 29	26 27	25 27
46 47	30 31	30 31	30 31	30 31	29 31
48 49	48 49	40 41	36 37	34 35	33 35
50 51	50 51	42 43	38 39	38 39	37 39
52 53	52 53	44 45	44 45	42 43	41 43
54 55	54 55	46 47	46 47	46 47	45 47
56 57	56 57	56 57	52 53	50 51	49 51
58 59	58 59	58 59	54 55	54 55	53 55
60 61	60 61	60 61	60 61	58 59	57 59
62 63	62 63	62 63	62 63	62 63	61 63

Present the table to a friend and ask him or her to determine which columns contain his or her age. For example, if your friend says his or her age can be found in columns 2, 3, 5 and 6, without searching the table yourself, you can determine that the age is 27 in a second.

Isn't that amazing?

The trick is that the age can be determined by adding the numbers at the top left-hand corners of the columns that contain the age. For example, the numbers at the top left-hand corners of columns 2, 3, 5, 6 are 16, 8, 2 and 1 respectively. Therefore, the age is the sum of 16, 8, 2 and 1, which is 27.

Can you work out how the table is constructed? (Hint: The principle is related to the binary number system.)

The von Neumann architecture

John von Neumann, a Hungarian mathematician, proposed an architecture of general-purpose computers in 1945 (see Figure 1.7). His proposal included the following ideas:

- 1 There are four main components in a computer. They are the **Arithmetic and Logic Unit (ALU)**, the **Control Unit (CU)**, **Memory**, and **Input/Output devices**:

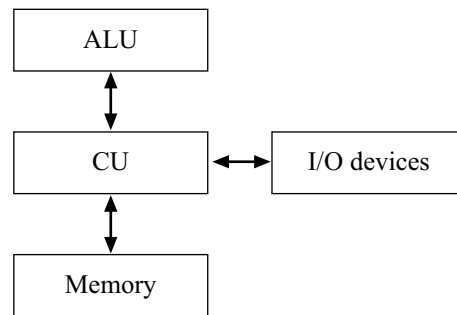


Figure 1.7 The von Neumann architecture

- a The Arithmetic and Logic Unit (ALU) is responsible for numerical and logical calculations.
 - b The Control Unit (CU) controls the sequence of operations to be carried out and coordinates the components of the computer systems to perform the tasks according to the stored program.
 - c Memory stores the data to be processed, the intermediate and operation results, and the program instructions.
 - d Input/Output devices interact with the users to get input data and present the operation results (output).
- 2 The control unit (CU) fetches one instruction at a time from the memory, then decodes it and executes it. This is called the Fetch-Decode-Execute cycle.
 - 3 Program instructions are encoded in binary format and stored in memory.

The idea of the von Neumann architecture has been implemented in all general-purpose computers. The four components can now be found in all computers.

Central Processing Unit and computer systems

The **Central Processing Unit (CPU)** is the brain of a computer where operations take place. It is similar to a calculator that performs calculations but at a much higher speed. The CPU is composed of ALU, CU and registers. Registers are fast memory that store data to be used immediately by the ALU or that have just been produced by it. To have a complete computer system, memory and input/output devices are needed.

A simplified diagram of a computer system is shown in Figure 1.8.

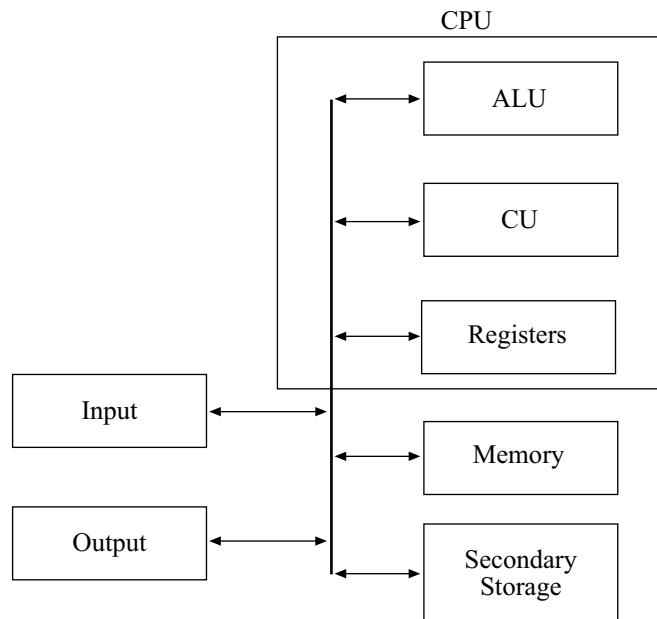


Figure 1.8 A simplified computer system

Notice the components of the computer system are the same as those in the von Neumann architecture, except that the memory of the von Neumann architecture not only refers to the memory that stores data and programs, but also the registers. The **secondary storage**, such as a hard disk, is just an input and output device. It is now obvious that von Neumann architecture is the basis of modern computers.

In a computer, a special register known as the **instruction pointer** (IP) or **program counter** is used to store the location of the next instruction to be executed. The Control Unit fetches an instruction from the memory according to the instruction pointer, and increases the contents of (increments) the instruction pointer to the location of next instruction. The Control Unit then decodes the instruction and executes it.

As discussed earlier, early computers or CPUs were built with vacuum tubes that had high-power consumption rates and needed frequent replacement. Later, transistors enabled the CPU to execute faster and consume less energy. Modern CPUs all use Integrated Circuitry (IC) that allows a single chip containing many small transistors to operate in it. Also, a special circuit connects all components in a CPU for transferring data between different components. We call this special circuit a *system bus*.

Memory

A computer needs storage space for both programs and data when it executes a program. Such storage space is known as memory or *primary storage*, which can further be divided into two categories — **Read Only Memory (ROM)** and **Random Access Memory (RAM)**.

Read Only Memory

As the name indicates, ROM is read only, and you cannot change its contents. Contents stored in ROM will not disappear when the power is off. It is still there when you switch on the computer. ROM is usually used to store small utility programs and some basic programs for accessing various components of the computer, and the data for initializing a computer and loading the programs necessary for the start-up process.

Random Access Memory

RAM stores data that can be changed at any time and into any pattern. However, the contents of RAM need a continuous power supply to maintain them. Therefore, when the power is switched off, all RAM contents are immediately lost and cannot be restored when you switch on the computer again.

Memory size measurement

To measure the memory, program or data size, there is a standard measurement convention. A '0' or '1' is known as a **BI**nary digiT, or simply a **bit**. A group of eight bits is called a **byte**. A single byte represents a digit, a letter of the English alphabet or some other symbol. The following table shows the units for measuring memory size:

Unit	Equivalent to	Data represented
bit		either 0 or 1
byte	8 bits	a letter or symbol
Kilobyte (KB)	2^{10} bytes = 1,024 bytes	about half an A4 page of text
Megabyte (MB)	$1024K = 2^{20}$ bytes = 1,048,576 bytes	about the text in your textbook
Gigabyte (GB)	$1024M = 2^{30}$ bytes = 1,073,741,824 bytes	
Terabyte (TB)	$1024G = 2^{40}$ bytes = 1,099,511,627,776 bytes	

Although the above units are standardized in the computer industry, most hard-disk manufacturers use the conventions that 1 MB and 1GB are considered to be 1,000,000 bytes and 1,000,000,000 bytes respectively.

Peripherals

Computers need some external devices for getting requests or comments from the users and presenting the operation results to them. According to the functionalities of the devices, they can be categorized into input, output and storage devices. Furthermore, some **peripherals** act as an input as well as an output device.

Input devices

Input devices receive the data for processing. Popular and basic input devices include the keyboard and mouse. Other optional input devices are the joystick, microphone, digital camera and image scanner.

Keyboards are designed based on typewriter keyboards. The key arrangements are probably in QWERTY. Because of the requirements of computer input operations, some special keys are added such as the Control, Alternate and arrow keys. Keyboards might also provide a numeric pad enabling input of numbers in a more convenient way.

The early computers were mostly text-based, and users were required to remember a lot of commands for operating the computers. Computers became more user-friendly after the introduction of the Graphical User Interface (GUI) and mouse. Nowadays, users rely more on a mouse. Try to think whether you can operate your computer without a mouse. Do you know how to shut down your computer properly using your keyboard?

A joystick is a common input device for playing games. For motor racing games, a car controller with a steering wheel, accelerator and brake pads can give you even more fun.

A microphone enables you to input audio signals to the computer so that it can digitize the analogue audio signal into digital formats. The computer can then further process and store the digitized signals in a storage device.

A digital camera or digital video recorder can capture still or moving images. If the digital camera is connected to a computer, you can have a videoconference with friends or business partners who reside in a remote location, if the computer is equipped with proper software and connected to the Internet.

Output devices

Output devices enable a computer to present the operation results. You might need different output devices for different types of output result.

Monitor

A **Visual Display Unit (VDU)** usually refers to the monitor of a computer. Nowadays, the technologies used for manufacturing monitors are mainly Cathode-Ray Tube (CRT) or Liquid Crystal Display (LCD). CRT is used for manufacturing television sets — it is a mature technology and the cost is relatively lower than that of LCD. However, CRT technology restricts the monitor from getting smaller and reducing its power consumption. LCD monitors are lighter and consume less electrical power than CRT monitors do, but they are more expensive. Some manufacturers design and make head-mounted goggle-like displays. Some head-mounted displays can track the head movements, and the output of the display is changed according to the head movement. When the user looks at such a VDU, he or she feels he or she is in another environment. Such technology is known as **Virtual Reality**.

Output generated by the VDU is volatile, because it disappears when the power is off. To get a hard copy of the output, you need a printer.

Impact printers

Early printers mostly used physical impact to generate the output. The first generation of impact printers included golf-ball and daisy wheel printers. Golf-ball printers used a spherical metal ball with different characters on its surface. To print a character on a piece of paper, the printer rotated the ball to the proper orientation and struck the piece of paper with a colour ribbon in between. A daisy wheel printer used a wheel with characters on its circumference. To print a character, the printer rotated the wheel so that the desired character was on the colour ribbon. Then, the printer used a hammer to strike the character on to the colour ribbon, and hence the character image was transferred to the piece of paper.

A problem with the above-mentioned impact printers is that it was not possible to print characters with a different typeface unless the golf-ball or the daisy wheel was changed.

Later, another type of impact printer became available — the dot-matrix printer. This printer has a group of small hammers, usually 9 or 24. A character is made up of a matrix of dots. An example character of a 9-pin matrix printer is shown in Figure 1.9.

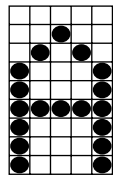


Figure 1.9 A dot-matrix image

Such a printer could print characters in different styles and typefaces.

A deficiency of all impact printers is the operational noise. If you need a quiet operation, you can use another category of printers that are non-impact, such as laser printers and ink-jet printers.

Non-impact printers

Laser printers use the same technology as common photocopiers. A laser printer receives the data to be printed and uses a laser beam to produce the desired image on a drum. Wherever the laser beam hits the drum, the electrical charge there is altered. The drum then rolls through a reservoir of toner, and the parts of the drum with altered electrical charge pick up toner. The toner is transferred to a piece of paper. By heat and pressure, the image is permanently printed on the piece of paper. Most laser printers contain one drum and can only generate black-and-white printouts. There are laser printers that have more than one drum, each responsible for a different colour so that they can generate colour printouts.

Another common type of non-impact printer uses ink-jet technology. Ink-jet printers have nozzles and emit ink when they pass over a piece of paper.

There are printers using other printing technologies, such as solid ink and dye-sublimation, which are usually used by specialists in the printing industry.

Storage devices

To cope with the fact that RAM contents will be lost once the power is disconnected, separate storage devices (known as secondary storage) are used for storing data permanently.

Since a computer processes data in strings of 0's and 1's, the storage media must be able to support two states so that it can identify the data as 0 or 1. Current storage devices are usually magnetic or optically based.

Magnetic-based storage devices

Magnetic-based storage devices include floppy diskettes, hard disks and digital tapes. The technologies used are basically the same. The surface of the storage medium is coated with material that is sensitive to magnetic fields. The surface is considered to be an array of dot positions. When the data are written to the magnetic surface, the read/write head modifies the polarization of the dot position to represent either 0 or 1. To read the data stored, the read/write head detects the polarization and returns the data.

The physical dimensions of the storage media can limit the way they are accessed. The storage media of both floppy diskettes and hard disks are a disk and the read/write head that can find any location on the disk at any time. Therefore, both floppy disks and hard disks enable random access to the stored data. For digital tapes, however, as the data are stored sequentially on the tape surface, it is necessary to advance the tape until the start of the desired data is located and commence reading the data from that point. Therefore, tapes only support sequential read/write operations of data and are commonly used for backup.

Optically based storage devices

Another mainstream storage technology is optically based. A typical example is a CD-ROM (Compact Disc — Read Only Memory). As the name indicates, the contents of a CD-ROM are read only.

Compact Disc

Compact Discs (CDs) have been around since 1980. They were originally used mainly for storing audio data to be played on a CD player. After the launch of audio CD, the same technology was used for storing digital data.

Data are stored sequentially in a single spiral track, which starts at the centre of the disc and wraps around until it reaches the outer edge of the disc. On a single spiral track, there are pits and lands, as shown in Figure 1.10. The CD reader reads the data on the track by firing an infrared laser onto the CD-ROM. The reflected light is different for pit and land so that the reader's sensor generates the string of 0's and 1's accordingly. A single CD-ROM can store up to 650 MB of data.

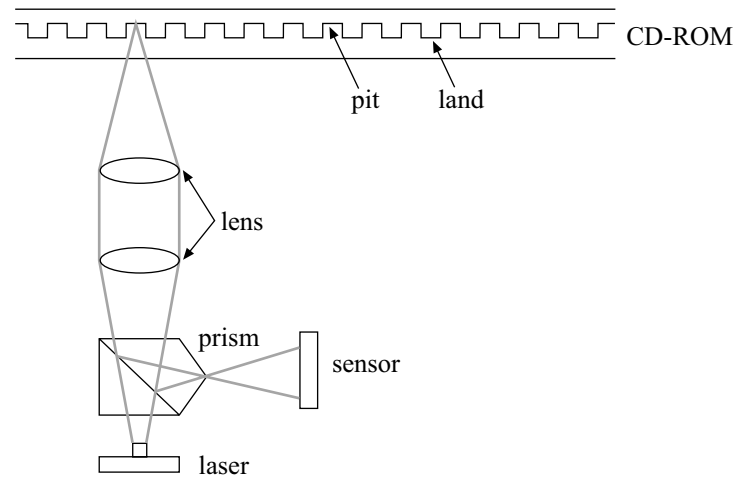


Figure 1.10 Simplified CD-ROM technology

There are two other similar storage devices — CD-Recordable (CD-R) and CD-Rewritable (CD-RW). Although there is no significant difference in physical appearance, they use different technologies.

Digital Versatile Disc (DVD)

Digital Versatile Disc (DVD) has a much greater storage capacity. It can store at least seven times more data than a CD-ROM can. Several DVD formats are available. Among them, DVD-Video and DVD-audio formats are used to store motion pictures and audio data respectively. DVD-ROM, DVD-R (DVD-Recordable) and DVD- Rewritable (or DVD-RAM) correspond to the functionalities of CD-ROM, CD-R and CD-RW respectively.

Others

There are also other secondary storage devices, such as memory stick and flash memory, which are used mainly for digital devices such as digital cameras and PDAs. By installing the memory stick reader to a computer, the pictures taken can be transferred to the computer for further image processing.

Self-test 1.3

Assume that a page of an ordinary document contains 250 words with an average word length of five letters. Calculate the number of pages that a CD-ROM can store (please note that each word is usually followed by a space).

Software

If you want to build a computer yourself, you can purchase suitable hardware components and connect them. Assume all hardware components are functioning and the connections are good. If you switch the newly connected computer on, you will find that it stops and the monitor displays an error message. What's wrong? When you look into the details, you can find that a small program stored in its ROM displayed the error message when it cannot load an operating system (see below for explanation). Without this software, the system just hangs and displays nothing. A computer does not work without software and is not useful without an operating system. Then, what is software?

Software is the program with the data necessary for the computer to operate on. Hardware and software must agree with each other so that the computer can do something useful. Software can be classified into operating systems and **application software**.

Operating systems

When you switch on the computer, a small program stored in ROM is executed to load a 'program' from the hard disk, which is the operating system (OS). For typical home personal computers, the operating systems can be Microsoft Windows 2000 or XP, or Linux.

An operating system manages and coordinates the computer resources such as the memory and file systems. Some operating systems also provide security features. For example, it can prompt the user for the username and password for verification and keep track of the user operations. Furthermore, an operating system typically includes some utility programs that can help the users in operating the computer.

Computer systems used to be expensive. If many users needed to use a computer, it was a good idea to share this expensive resource. Therefore, some operating systems were designed to support multi-users to access it and perform different tasks at the same time. Such operating systems are known as multi-user operating systems. A typical example is UNIX. Linux and FreeBSD are UNIX-like operating systems that also support multi-users.

The Microsoft Windows series operating systems provide a GUI so that the computer is more user-friendly.

Application software

The operating system just provides some basic utility programs for operating the computer. If you need to use the computer to perform some specific tasks, such as preparing a spreadsheet, you need application software. Open Office and Microsoft Office are examples of *suites* of application software.

Application software needs to match the operating system so that it can work properly. For example, it is not possible to execute Microsoft Office directly in Linux — you must use Microsoft Windows (or some emulation software) even though the hardware is the same, because application software cannot manipulate the hardware directly; it must access them by using the services provided by the operating system. Different operating systems provide the services differently. You can visualize the scenario by Figure 1.11:

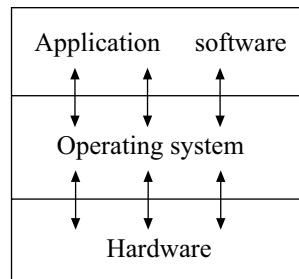


Figure 1.11 Separation of application software from the hardware

You can see that application software never accesses the hardware directly but sends a request to the operating system. The operating system accesses the hardware on behalf of the software application.

Programming languages

Language is a common predefined way to present ideas. For example, you communicate with your friends by saying something in a human language, such as Cantonese or English. If you want to communicate with the computer and instruct it to perform tasks, you need computer programming languages.

We use **programming language** to write programs or develop software so that when the computer executes the software, it performs the desired operations for us. Although different software packages are available in the software market, they might not fully fulfil your specific needs. Therefore, you need to learn programming languages so you can design and develop software for yourself.

There are many programming languages. They can be grouped into **low-level** and **high-level programming languages**.

Low-level programming languages

Machine languages and **assembly languages** are known as low-level programming languages, because their formats are closely related to the hardware that executes them.

Machine languages

Machine language is considered the first generation of programming language. Each CPU is designed with a machine language composed of 0

and 1 only. The CPU can only understand and execute programs written in such a machine language. For example the following can be an instruction to add the numbers 1 and 2:

```
00010001 00000001 00000010
```

The first group of bit patterns (00010001) means to add the two bit patterns (00000001 and 00000010) following it. To execute machine instructions, the CPU interprets the bit patterns and carries out the associated operations. Although it is hard for us to understand the above sequence of 0's and 1's, the CPU can easily match the patterns and operate accordingly.

Assembly languages

You cannot easily understand sequences of 0's and 1's, but how about the following:

```
ADD    1, 2
```

There is an English word, ADD, in the instruction (or statement) as a mnemonic. You probably can guess the instruction is to calculate the sum of 1 and 2. Then, the CPU will store the result in a temporary location, a register, so that it can use it for further processing.

Language in the above format is known as assembly language. Each assembly language instruction is converted into a single corresponding instruction in machine language. Since a CPU has one machine language, it has one assembly language. For example, the above instruction might be translated into the following sequence of 0's and 1's for a particular CPU:

```
00010001 00000001 00000010
```

Then, the computer can handle the sequence of 0's and 1's to perform the desired task. Assembly language is easier for us to write and understand. It minimizes the requirement for memorizing the patterns of 0's and 1's and provides a one-to-one mapping from assembly language instruction to the corresponding machine language.

Assembly language is also machine dependent, because the translated sequences of 0's and 1's in machine language can only be understood by the target CPU. Assembly language is usually referred to as the second-generation language.

High-level programming languages

If you have written a program for a particular CPU using its assembly language, and you later want to enable the users who are using another CPU to use the same program, you have to rewrite the program in another assembly language. Such a process is time-consuming because you have to learn another assembly language and rewrite the program from scratch.

High-level language enables us to develop software for different computers without or at least with much fewer modifications.

Can you understand the following statement?

```
IF NUMBER > 0 THEN PRINT "POSITIVE"
```

The above statement is typical in the high-level programming language, **BASIC**. It is relatively easy to understand what the statement means. The statement checks if the value stored in **NUMBER** is greater than zero and prints a message 'POSITIVE' if that is true.

You can see that a feature of high-level programming languages is the human language-like statements that are much easier to understand and use.

High-level programming languages can use descriptive names for storing values, just like the **NUMBER** in the above statement. It does not stick to a particular platform (the combination of CPU and operating system) and therefore can be executed on different machines with or without minor modifications.

One of the first high-level programming languages was **FORTRAN**, created in 1954. Since then, other high-level programming languages were created such as **COBOL**, **BASIC**, **Pascal**, **C**, **C++** and **Java**. These programming languages are considered third-generation languages.

There are fourth- and fifth-generation languages. Compared with third-generation languages, fourth generation languages (4GL) are even closer to human languages. For example:

```
SELECT NAME FROM STUDENTS WHERE STUDENT_ID='0256820';
```

4GL are usually used for accessing databases. The above statement is written in a 4GL known as **Structured Query Language (SQL)**.

Fifth-generation languages support declarative programming. The computer is able to deduce results based on the facts and rules inputted. Such programming languages are usually used for developing software that supports Artificial Intelligence.

We mentioned that a computer only understands its native tongue — machine language — so a program written in a high-level programming language must be translated into the corresponding machine language instructions so that the computer can execute them accordingly.

Comparisons between low-level and high-level languages

The following table summarizes the differences between low-level and high-level programming languages:

	Low-level programming language	High-level programming language
Speed	Faster because the programmer can make use of the registers directly and avoid redundant operations	Slower because the computer needs to translate the high-level language to machine language for execution, but the machine code generated may not be optimal.
Portability	Impossible	With no or minor modifications.
Readability	Difficult to read and modify	Much easier to learn, write and modify.
Selectivity	No choice. Each machine supports its own machine language	A computer can execute programs written in any high-level programming language, provided that a translation software is available.

Reading

King, Section 1.5, pp. 15–17; stop after reading the ‘History of Java’.

How programs are executed

There are two methods to execute programs written in high-level programming languages (known as **source code**). The first one is to convert the source code to the machine language of the computer (known as **object code** or **machine code**) and to execute the machine code that is supported by the computer hardware. This conversion process is called **compilation**. The second one uses a software interpreter, which is already in the machine language of the computer, to execute the source code directly. This is called **interpretation**.

Interpretation

When you run a program written in high-level programming language with interpretation, you execute a software interpreter that interprets (analyses) each statement in the program and then carries out the operations one after another. Some interpreters translate the high-level instructions to a form that is easier to interpret.

Notice the similarity between hardware interpretation by the CPU and the software interpretation by the interpreter. Both analyse their own instructions and carry out the associated operations.

Some programming languages are natively implemented with interpretation only, such as **Perl** and **JavaScript**. An advantage of executing a program with interpretation is that the interpreter software starts running the program after it reads the first instruction, which is different from what compilation does.

A shortcoming of interpretation is that the high-level instructions need to be analysed each time they are executed. If an instruction is executed repeatedly, the interpreter software repeats the analysis of the same instruction for execution. In the following example program written in the BASIC programming language, the computer is instructed by the program to repeatedly display the message 'HELLO WORLD' 1000 times. This is done by first assigning a value of 1 to a variable `I` (line 1), and then repeating the printing (line 3) and incrementing actions (line 4, increase `I` by 1) until `I` is greater than 1000.

line	statement
1	LET I = 1
2	REPEAT
3	PRINT "HELLO WORLD"
4	LET I = I + 1
5	UNTIL I > 1000

The second to fifth lines of the program are analysed 1000 times during the program execution.

Compilation

If a program written in a high-level programming language is entirely translated into the corresponding machine language before execution, it is known as **compilation**. The software that compiles a program is compiler software (or simply a compiler). The result of compilation is known as object code, which is composed of 0's and 1's. The machine only executes the object code. The source code is no longer needed until there is a need to understand the program logic or modify the program behaviour. The process is summarized in Figure 1.12.

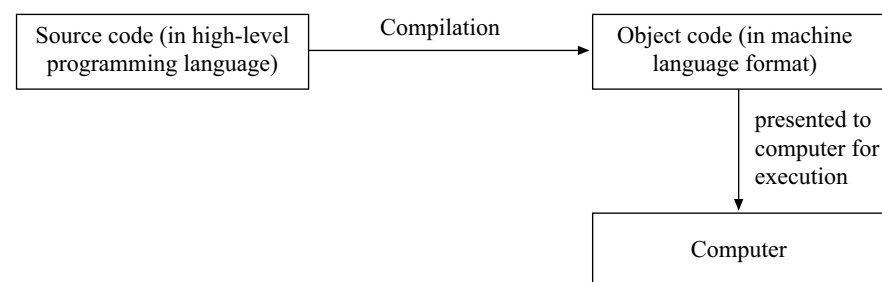


Figure 1.12 Compilation and execution

The notable advantage of compilation is its high execution speed. If you were going to perform a computation-intensive task such as scientific simulations, it would be better to execute the program with compilation rather than interpretation.

The object code generated by compilation is in the format of machine language, which is a string of 0's and 1's. Therefore, it is difficult for others, even the original programmer, to understand. This is very important for software development companies, however, because the users would then not be able to determine the logic of the software, let alone the software, and claim ownership.

The first drawback of compilation is that it needs a compilation process before execution, whereas interpretation can start executing a program immediately. Furthermore, the result of compilation is in a machine-language format. As a result, it is necessary to compile the original program that was written in high-level programming language once for each machine type.

The following table compares the various processes of compilation and interpretation:

	Interpretation	Compilation
Execution model and speed	Slower, as every statement is analysed each time before execution.	Faster, as object code is generated once and for all. No translation during execution.
Execution model	Needs source code for real-time translation.	Source code is not necessary — just object code.
Portability	Needs an interpreter on each platform.	Needs compilation once for each platform.
Reliability	Usually less reliable, as checking is done during interpretation. For program segments that have not been executed, there is no checking for them.	More reliable, because compilation usually includes comprehensive checking for the whole program.
Intellectual property	Weak protection, since users can read and modify the source code.	Strong protection, since users find it difficult to derive the program logic or modify it.

The programming language Java, which you will learn in this course, uses partial compilation and partial interpretation. The compilation process translates the Java source code to Java bytecode (object code) that is not in a hardware machine-language format. The Java interpreter (also called the Java Virtual Machine) then executes the Java bytecode to carry out the instructions.

Reading

King, pp. 17–18. Then try the review questions on p. 18. The answers are on p. 26.

The Internet and the World Wide Web

A single computer can help execute a lot of operations. A few computers connected form a local area network (LAN) so that all members in the network can share some common resources such as printers. Roughly speaking, the network that connects a lot of such sub-networks is known as the Internet.

The Internet

Today, we connect our personal computers to the Internet for sending email, searching for text-based or multimedia resources and for playing games. However, the original aims of the Internet at its early stage were completely different.

The history of the Internet

On 4 October 1957, the then Soviet Union launched its first artificial earth satellite, Sputnik 1. Then, US President Dwight D Eisenhower determined that it was necessary to establish a research agency to investigate the application of science and technology in the military area. The Department of Defense (DoD) therefore set up the Defense Advanced Research Projects Agency (**DARPA**), formerly called the Advanced Research Projects Agency (**ARPA**).

The US Air Force wondered how they could control their nuclear weapons after a nuclear strike if any part of the US were attacked. The problem was presented to DARPA. The agency provided the solution of setting up a decentralized network. Information to be communicated on the network was broken down into small messages, known as packets, and each could take its own path to the destination and then be reconstructed to form the original information. If any one of the messages was lost, it could be resent. These were the underlying principles being used up to the current Internet.

In 1967, several researchers proposed a plan for a computer network that was known as the **ARPANET**, the ancestor of the Internet. The proposed network was built in 1969. The University of California Los Angeles (UCLA) was chosen to be the first node on the ARPANET. The second node was the Stanford Research Institute (SRI). Afterward, the University of California Santa Barbara (UCSB) and University of Utah (Utah) were added (see Figure 1.13).

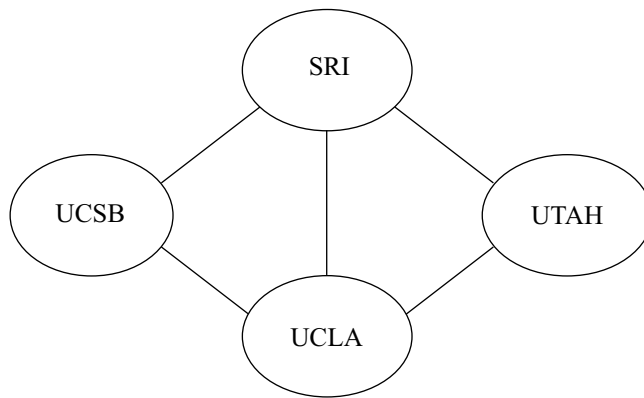


Figure 1.13 An early form of ARPANET

At that time, the ARPANET connected only four computers in four universities in the United States with circuits that supported a communication speed at 50 Kbps (kilo-bits per second). This is slightly slower than the current connection speed between home computers to the Internet with a 56 Kbps modem using the usual telephone line.

In the following years, many other computers were added to the ARPANET. The first public demonstration of the ARPANET was organized in 1972 at the International Computer Communication Conference (ICCC). In the same year, the popular application, **electronic mail** was introduced.

In 1973, international connections from England and Norway to the ARPANET were established. At the same time, several other similar networks were set up using different communication approaches for various academic and research purposes.

Also in 1973, DARPA started a research program to connect various networks. The purpose was to develop a communication standard that would allow various networks to communicate transparently. It was known as the Internetting program. The network systems and standards that emerged from it were known as the 'Internet'.

Common services

The original aim of creating the Internet was for academic purposes such as communication among researchers at different locations around the world. Advances in communication technology have brought about the possibilities for other uses of the Internet. Some common uses of Internet today are presented in the following paragraphs.

Email

Since the Internet came into existence, one of its major uses has been for sending and receiving email. In the early days, it was difficult for a person to obtain an email account and address. Nowadays, many Internet Service Providers (ISPs) provide free email services, and any user can apply for one. As a result, many Internet users have more than one email address.

FTP

File Transfer Protocol (FTP) is a means of sharing files across the Internet. By placing files in an FTP server, they can be accessed by authorized or anonymous users. A use of FTP can be for storing tutorial and assignment documents in a specific FTP server so that students can retrieve the files anytime from anywhere.

Telnet

Some computers are classified as time-sharing machines so that more than one user can connect to the machine through the network and perform their own operations. This is similar to the use of 'Command Prompt' of Microsoft Windows. **Telnet** can emulate a terminal to the machine so that more than one person can use the machine at the same time. Unfortunately, there is no graphical user interface. Users are required to enter commands from their keyboards in this text-based environment. Therefore, only relatively skilled computer users are able to use it.

ICQ

The creation of **ICQ** brought about a new era of communication on the Internet. ICQ, which stands for 'I-see-you', integrates several communication means including real-time chatting, off-line email, file sharing and so on. With proper peripheral devices, you can even send an electronic voice mail to target recipients.

To participate in the family of ICQ, you register with the ICQ service provider first and get an ICQ number that uniquely identifies you in the world of ICQ. Afterwards, you are ready to chat with and even make new friends who have joined ICQ.

The World Wide Web

In 1989, researchers at the European Laboratory for Particle Physics, also known as CERN, proposed a new communication standard for distributing documents. The standard was based on the use of embedded links (or hyperlinks) to other documents. This standard brought about the World Wide Web in 1991.

The first Web browser, Mosaic, was built by the National Centre for Supercomputing Algorithm (NCSA) and became publicly available in 1993. The team leader later left the NCSA and headed Netscape Corporation and developed the popular Web browser, Netscape.

Some people misunderstand that the Internet is equivalent to **World Wide Web (WWW)**. Actually, WWW is only one of the common uses of the Internet.

Hyperlinks, Web pages and websites

Since the Internet is a shared network that is composed of a lot of computers, it is an excellent means for sharing materials.

Some documents, such as research papers, usually refer to other research papers (listed in their reference sections). Traditionally, if the readers wanted to read the papers referred to, they had to find them manually, probably in the library. For easy access to referred documents, the original documents (called hypertexts) are specially edited so that these referenced items can be accessed by hyperlinks (embedded document locations).

The hyperlinks among documents form a web-like structure in the Internet. Therefore, such a repository of resources is called World Wide Web or simply WWW. The documents mentioned are also called Web pages. You only need a Web page browser, such as Netscape or Microsoft Internet Explorer, to read the Web pages.

Every machine on the Internet has a unique numeric network address, which usually has a corresponding mnemonic name for easy reference. We can use these names to locate a machine on the Internet. For machines that are running a Web server and storing Web pages, you can use a Web page browser to read the Web pages on them. For example, the Open University of Hong Kong runs a Web server and stores documents related to the University. You can present the following Universal Resource Locator (URL) <http://www.ouhk.edu.hk/> to a Web browser to read the Web pages available to the public via the Internet. The collection of Web pages provided by an organization or company is known as a website.

Java, what is it?

Have you heard the name Java? It is the name of a popular coffee, a street in Hong Kong, an island in Indonesia, and so on. In the IT industry, Java is the name of a new programming language that was launched in 1995 and has become one of the most important programming languages on the planet.

The history of Java started in 1991. The company, Sun Microsystems, was doing a project called the Green Project, which designed and manufactured handheld devices for home entertainment. They not only created the device but also a programming language that worked with the device. The programming language was named Oak, the name of a tree outside the language designer's home.

The Oak language was designed to be object-oriented (discussed in the following section). Programs written in Oak could work with various hardware. After the project, the designer of the language wanted to find the development direction of the language. In that year, because of the growing popularity of the Internet and World Wide Web, the development direction of Oak was changed to aim for the Internet. At the same time, the name of the language was changed to Java.

In a conference held in 1995, a **Java applet** was written to demonstrate its capability of running an application within a Web page. At that time, Web pages were static and no dynamic content was available. The Java applets surprised all conference attendants. The company released the language and the necessary tools at no charge. Its capability of being executed in various platforms has made it one of the most popular programming languages since then. In the same year, the first public release, version 1.0, was released.

Why object-oriented programming and Java were chosen for MT201

From the invention of computers up to now, many programming languages have been designed and implemented. Some languages are imperative, some are object-oriented and some are a hybrid of the two.

Modern computers are designed based on the von Neumann architecture discussed earlier. As the execution model of von Neumann architecture is *imperative* (discussed very soon), the early programming languages were naturally designed to be imperative as well. Later on, researchers found that modelling the real world was a better way to develop software. Programming languages that can model real-world entities are classified as *object-oriented*. Because many programmers already knew the imperative language C, object-oriented features had been added to it to form a new hybrid language, C++.

In this course, you'll learn programming in Java. It is a hybrid language containing object-oriented features at the class level and imperative features at the method level, which are quite similar to C++. Let's consider the reasons why they have been chosen.

The imperative programming paradigm

Imperative programming language is also known as procedural programming language. It manipulates data in a step-by-step way. A program written in an imperative language consists of commands, the purpose of which is to effect a change of either state or execution order. Programs written in imperative programming languages execute statements or instructions one-by-one and apply them to data.

Computer hardware implementations are basically imperative. As we discussed previously, a computer can only execute programs written in its own machine language. The computer executes the machine language instructions one-by-one; each instruction is applied to some data stored in either or both the memory and CPU registers.

As an enhancement of low-level programming language, most early high-level programming languages were also imperative. The differences were only that the statements were high-level instructions defined in the language and the data were specified by the variables defined in the program.

The C programming language is a typical imperative programming language that was originally designed by Dennis Ritchie at Bell Labs in the mid-1970s for developing the UNIX operating system. Since then, C has been used to develop software in various problem domains. However, the imperative programming languages have some serious limitations, discussed in the next section.

Because of the limitations, Bjarne Stroustrup, the head of AT&T Lab's Large-scale Programming Research Department, had added object-oriented features to the C programming language. The new language was called C++ in 1983; it is a popular object-oriented programming language (in addition to the Java programming language) today.

The pitfalls of the imperative programming paradigm

Imperative programming languages have been around for a long time, and many computer applications are written in such languages. However, it was found that imperative programming languages exhibit some common pitfalls that make them unsuitable for developing large-scale and complex computer software.

- 1 In every program, the programmer has to specify 'what' problem to solve and 'how' to solve it. The imperative programming paradigm emphasizes the detailed steps or the algorithms (how) for solving a problem. A programmer may need to spend over 80% of the time in the 'how' part. If you encounter a problem and want to solve it using

an imperative programming language, you might find that you are distracted because you have to design the solution as a high-level abstraction and low-level implementation at the same time.

- 2 There is little support for programming-in-the-large and team development. If a large software system is to be developed, there may be a lot of people involved. If the programming language does not provide facilities to minimize interactions among different parts and subparts of the system, people changing the code of a part may affect other parts of the system unintentionally. This makes system development and maintenance difficult.
- 3 Programs written in imperative programming language are mostly problem specific. There are no facilities for enhancing software reuse, and reusing written programs is not an easy task.

Why object-oriented programming?

Before we discuss why object-oriented programming was chosen in the course, let's consider what it is. First of all, please don't be scared by the term 'object-oriented'. You will later find that you are already familiar with the underlying concepts.

Let's consider a scenario (Figure 1.14) in which you order a pizza from a restaurant, over the phone. You make a phone call to the restaurant and place an order for the style and the size of the pizza and give the delivery address. The operator in the restaurant then sends a request with the style and size of the pizza to the kitchen for preparation. As soon as the pizza is ready, it is forwarded to the delivery team. A staff member there will get the address and the total price from the operator and send the pizza to your home. When it arrives, you pay and get the pizza. The delivery staff will take the money back to the office.

The key point in the above scenario is that it involves some people (or agents) and a few requests (or messages). The whole operation is initiated by your request to the operator. The operator subsequently sends a request to the staff in the kitchen. After the pizza is ready, a request is sent to the delivery team. The delivery team gets the delivery request and sends a request to the operator to get the address and the bill. A delivery person brings the pizza to you and asks for the payment.

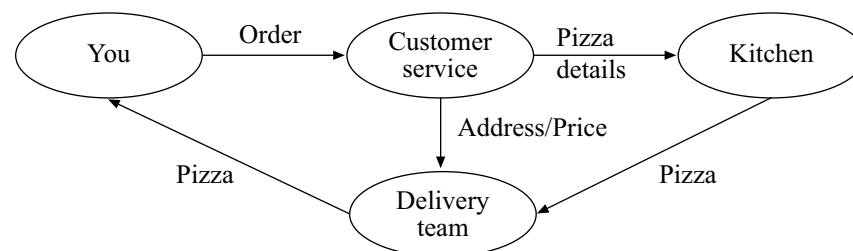


Figure 1.14 The scenario for ordering a pizza by phone

A key observation is that the people involved do not need to know the operational details of one another's work. It is just necessary for each of

them to perform his or her own operations and send requests to other people for subsequent actions. Everyone trusts one another that each person can complete his or her job.

You can easily find many more real-life examples that involve some agents who cooperate with one another through sending messages. It is a natural solution for handling a problem involving different people or agents.

From the perspective of writing programs, you can solve a problem by first determining the involved agents and the messages that are sent among them. Afterwards, you can concentrate on each agent and write programs to perform all the determined operations.

The agents mentioned above are known as *objects* in programming; such software development methodology is classified as object-oriented. Object-oriented programming languages can implement objects and enable message passing among objects.

Object-oriented programming was chosen for the course because it is a more natural methodology for solving problems, as you can see in the above scenario. Furthermore, you will learn later in the course that software developed using object-oriented programming is more robust, easier to debug and maintain, and it is easier to reuse the components that you have built.

Is Java an object-oriented programming language? Why Java?

The answer is both yes and no. Java is a hybrid language involving both the object-oriented and imperative features. It supports the main features of an object-oriented programming language, but it is not a pure object-oriented programming language because it still enables programmers to develop non object-oriented programs. It is therefore up to the developers to write programs with object-oriented methodology or not.

In computer science, many programming languages are purely object-oriented. However, they are less popular, and some implementations are quite limited.

Although Java is not a pure object-oriented programming language and its history is relatively short, it is one of the most popular programming languages. The following is a list of advantages of using Java over the others.

- 1 Java is a real programming language. Since the language was designed, software development kits (SDKs) have been made available. Such software development kits are free of charge, and you can find the kits for almost all operating systems. Currently, the SDKs for Microsoft Windows, Sun Solaris and Linux are officially supported by Sun Microsystems. The corresponding vendors for other platforms will provide their own SDKs.

- 2 Java is platform independent. A program written in Java can be executed in all platforms that have a proper SDK or Java runtime environment (JRE) installed, without modifying a single statement. Such capability is very important, because you do not need to assume the platform on which your program will be executed. Notice the SDK or JRE, which is in the machine language of the platform, includes a Java interpreter to run compiled Java programs.
- 3 Plenty of development tools are available. Many tools are available for developing **Java applications**, and most of them are either open source or free of charge. Currently, you can develop almost all software for different models and for different devices ranging from mobile phones to enterprise level supercomputers.
- 4 Java has a large user community, and the language is enhanced rapidly. Moreover, as the organization that created the language, Sun Microsystems maintains the language specification and monitors the language development, so that there are no variations and programmers do not need to cater for any differences.
- 5 The syntax of Java is more restricted than that of some other languages. Although it is more difficult to write a Java program that can be compiled, the resulting program is more reliable and robust.
- 6 Java was developed for use in the Internet. Being a language for the Internet, the Java programming language naturally supports security by various means. Therefore, it is much safer to use Java to develop applications for the Internet.

If you come across other programming languages later, you will appreciate the beauty of the Java programming language even more.

Although Java is not a pure object-oriented programming language, we use Java as a means to learn object-oriented programming methodology.

Reading

King, Section 1.6, pp. 18–19. Try the review questions on p. 19; the answers are on p. 26.

Java applications and applets

The Java programming language is well known for its ‘write once, run everywhere’ feature. You develop your software application with the Java programming language, and you can be sure that your software can be executed on all popular platforms, including Microsoft Windows, Linux, Sun Solaris, Macintosh and so on.

Also, you can develop your software in different execution models with the Java programming language. The execution model to be used

depends on the target users of your software and hence the way in which you would prefer the target users to use your software.

One of the execution models is a Java application that you can execute as a standalone software application. With such a model, it is necessary to install a Java Runtime Environment (JRE) and the compiled program in Java bytecode to the user computers, so that the users can start and execute the software. The shortcoming of this execution model is that every time the software is updated, you need to re-install the new software files to the user computers.

Another execution model is the Java applet. In the conference when the Java programming language was first demonstrated to the public, the software shown was built in the Java applet model. To execute a Java applet, the only tool the target users need is a Web browser. Most Web browsers are equipped with a Java Runtime Environment. Whenever a user uses the Web browser to view a Web page that embeds a Java applet, the Web browser will download the related software files onto the user's machine and run it immediately.

You can browse the following Web page with your Web browser to see some demonstration Java applets: <http://java.sun.com/applets/>

An advantage of Java applets is that when the software files of the Java applet have been updated, the Web browser will detect the changes and download the modified software files automatically. It greatly minimizes the workload to update the software files on the user computers. However, running software that is downloaded from the Internet can pose a serious security threat. Therefore, the Java Runtime Environment for running Java applet is specially customized so that the Java applet cannot perform operations that are potentially a security threat, such as opening a file stored in the computer hard disk.

Software written in the Java application model has no operational restrictions, and a Java applet eases software installation and upgrade. To take advantage of these two models, a new technology named Java Web Start was invented so that the users can start a Java application by clicking a hyperlink in a Web page or an icon on the Windows desktop.

Besides Java application and Java applets, other Java development models such as Java Servlets and Enterprise Javabeans (EJB) were invented that target different software markets.

MT201 concentrates on writing Java applications. After you have become familiar with the Java programming language, you can easily pick up the skills for developing Java applets, Java Servlets and so on.

Installation of Java Software Development Kit

To this point in the unit, you have come across the following terms (packages). You might be unclear about their differences. They are:

- 1 Java Virtual Machine (JVM)
- 2 Java Runtime Environment (JRE)
- 3 Java Development Kit (JDK) or Java Software Development Kit (Java SDK).

Table 1.1 summarizes the differences in these terms:

Table 1.1 Differences among Java packages — JVM, JRE, and JDK/SDK

	JVM	JRE	JDK/SDK
The Java Virtual Machine emulator (interpreter)	Yes	Yes	Yes
The standard Application Programming Interface (API) library	No	Yes	Yes
The compiler and other supporting tools	No	No	Yes

The JVM only refers to the emulator that can simulate a virtual machine for executing Java bytecodes. If you want to execute the compiled Java programs in bytecode format, the emulator needs the supporting standard application programming interface (API) library. Therefore, it is necessary for you to install at least a JRE in your machine. To develop Java programs, your machine must have JDK (or SDK) properly installed.

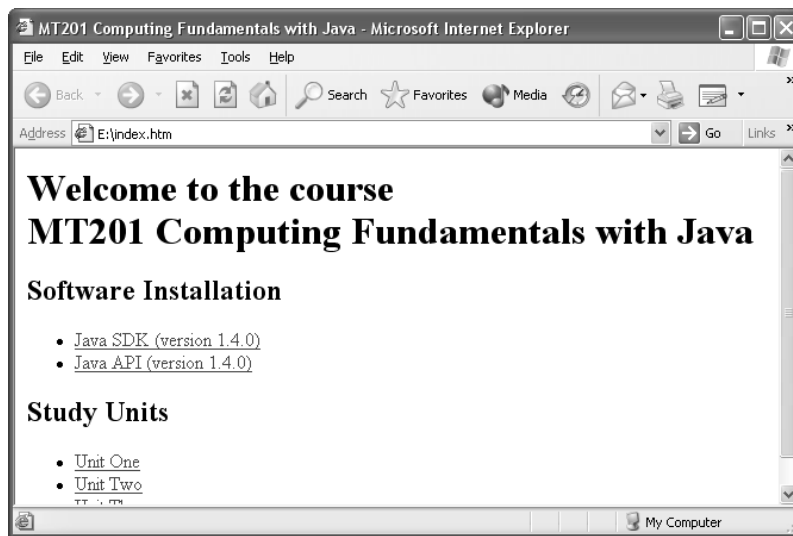
For the platforms that are officially supported by Sun Microsystems, which are Sun Solaris, Linux and Microsoft Windows, you can always download the latest versions of the JRE and SDK from the Java official website at: <http://java.sun.com>

You may need to look for J2SE SDK (Java 2 Standard Edition Software Development Kit). The size of the file to be downloaded is quite large, and it is almost impossible to download the files with a dialup connection. We have placed the Java SDK version 1.4.0 for Microsoft Windows in the course supplementary CD-ROM for your installation.

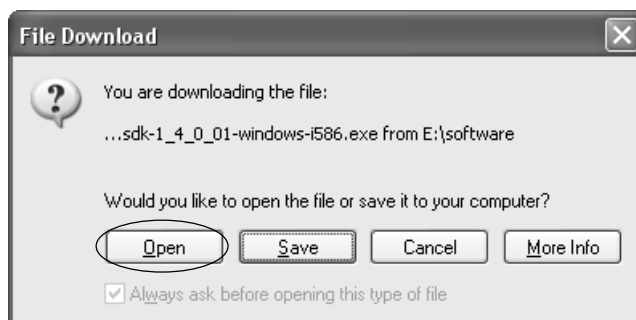
The unit only provides the installation procedure for Microsoft Windows with English as the native language. If English is not the native language of your machine, you should be able to find the equivalent items as mentioned in the steps. The installation process is basically the same.

To install the Java SDK in your machine that runs Microsoft Windows, you can follow the steps below:

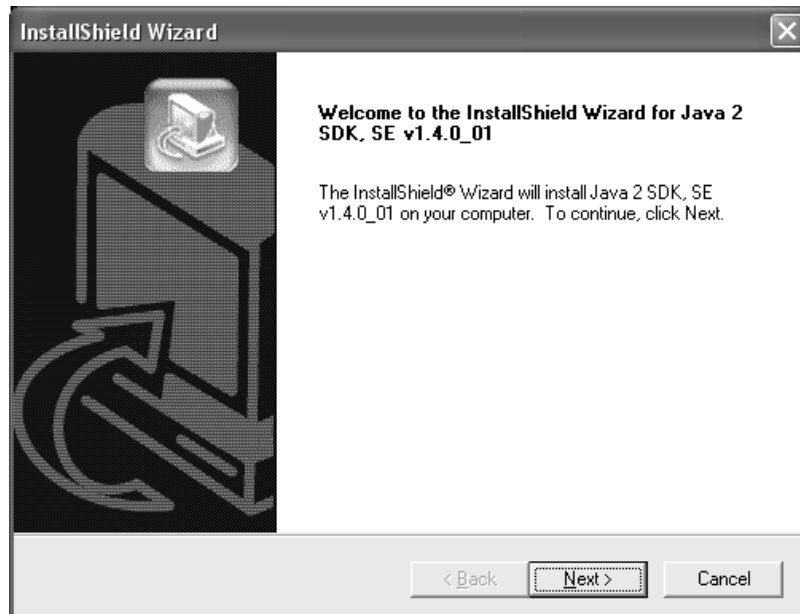
- 1 Insert the CD-ROM in the CD-ROM drive of your machine. The Web browser will start. (We are using version 1.4.0 as an example; the version of your copy may be different.)



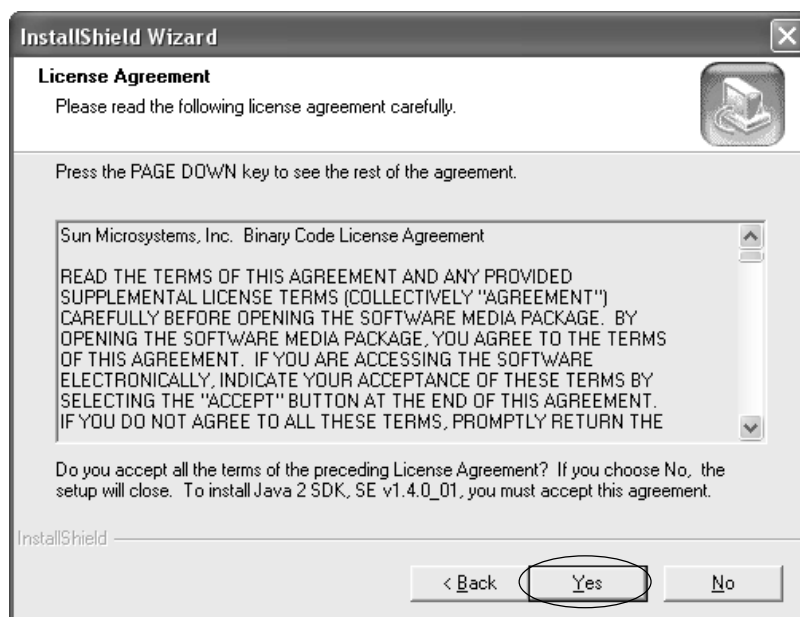
- 2 Click the link, [Java SDK \(version 1.4.0\)](#). You will be prompted with the **File Download** dialog. Then, click **Open** in the **File Download** dialog. The Java SDK installation process is started. (If the process does not start, double click the file j2sdk-1_4_0_01-windows-i586.exe.)



- 3 The **InstallShield Wizard** dialog appears. Click **N**ext to proceed. (We are using version v1.4.0_01 as an example; your version may be different. In the steps that follow, remember to replace '1.4.0_01' next to the version number of your copy.)

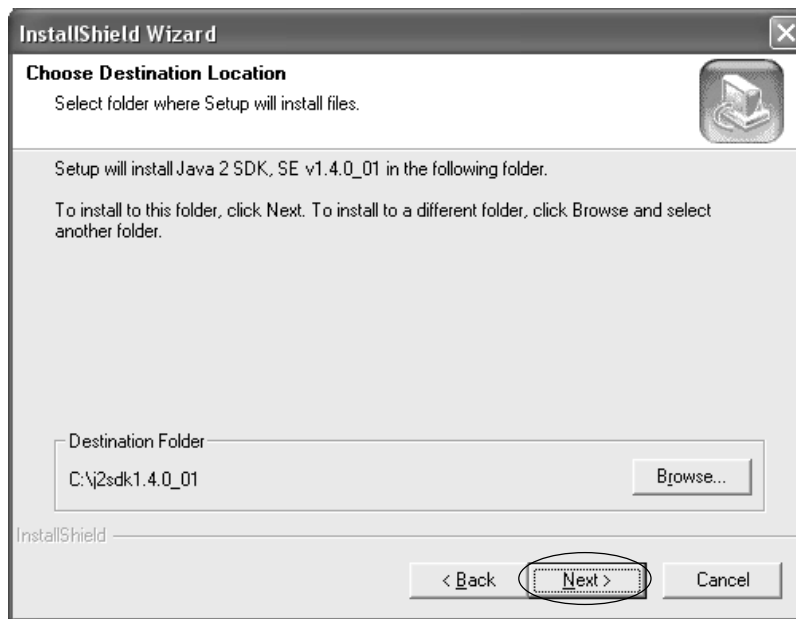


- 4 The dialog will show you the License Agreement that you have to agree to before continuing the installation:



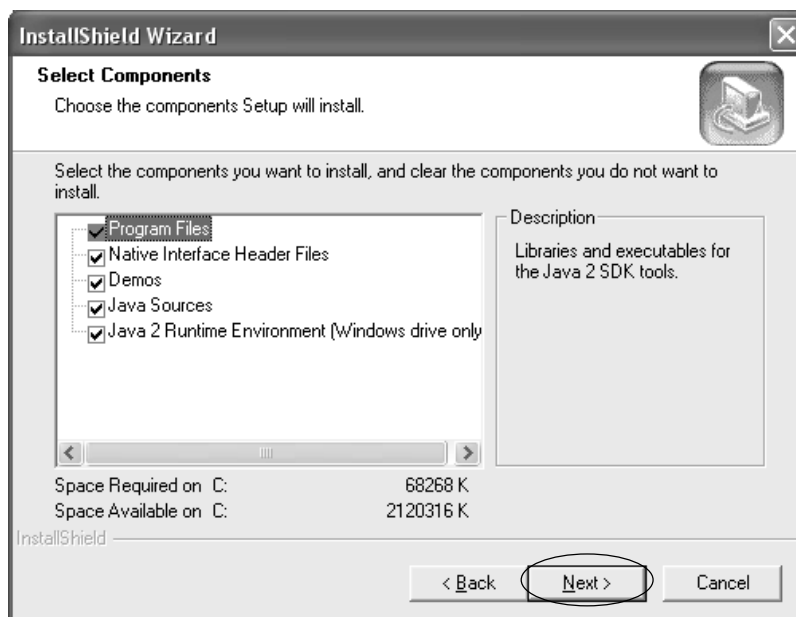
Press the <Page Down> key to see the whole agreement. After reading the licence and accepting the agreement, click **Yes**. If you click **No**, the installation process will terminate.

- 5 Select the destination folder for storing the Java SDK.



The default directory is 'C:\j2sdk1.4.0_01' (or a similar path depending on your version). It is recommended you keep this setting, because it is the default. The instructions in the unit are prepared on the assumption that you are using the default settings. If you have a special reason for changing the destination folder, you can click **Browse** to change the setting. Otherwise, click **Next**.

- 6 The dialog shows you the components to be installed in your machine. You can simply click **Next** to accept the default settings and proceed.



- 7 The Java SDK can install a plug-in to the Web browser that exists in your machine so that it can execute Java applets. The installation program will search your machine for installed Web browsers. You can keep the default settings by clicking **Next**.



- 8 The installation process continues, and it starts copying the necessary files to your machine. At the end of the process, the following dialog will appear to signify the end of installation.



- 9 Click **Finish** to close the dialog.

Congratulations! The Java SDK has been successfully installed in your machine. However, you still need to modify a few settings in your Microsoft Windows, so that you can use the SDK conveniently.

For Windows 95/98 or ME, the steps are:

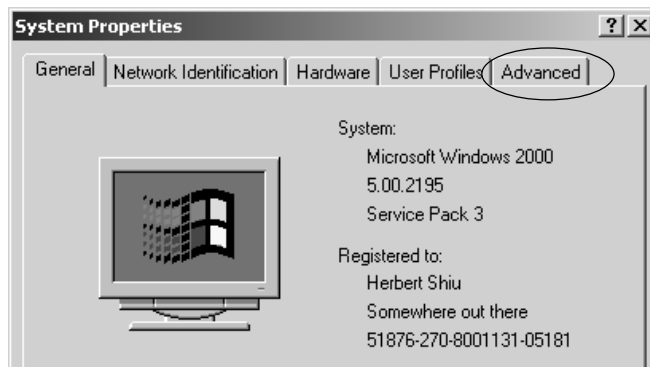
- 1 Select '**Start**' in the Windows menu. Then, select '**Programs**', '**Accessories**' and finally '**Notepad**'.
- 2 Select '**File**' of the **Notepad** application. Select the '**Open**' menu item.
- 3 In the '**File name:**' text box, enter '**C:\autoexec.bat**'.
- 4 At the end of the file, append a line as (the part 'C:\j2sdk1.4.0_01' should be the same as the directory of your Java SDK installation):


```
SET PATH=%PATH%;C:\j2sdk1.4.0_01\bin
```
- 5 Select '**File**' of the Notepad application. Select the '**Save**' menu item.
- 6 Close the Notepad application by clicking the close button at the top right corner.
- 7 Re-start your machine to ensure the settings become effective.

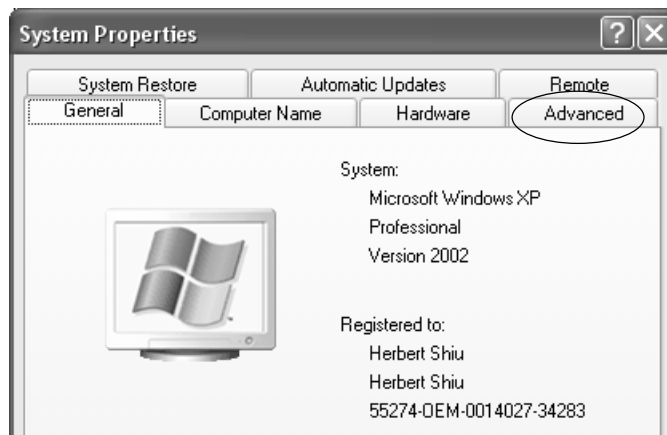
If your machine is running Windows NT, Windows 2000 or Windows XP, the steps are:

- 1 Right click **'My Computer'** on the desktop. Select the menu item **'Properties'**. If you are running Windows XP, you probably cannot find the **'My Computer'** icon on the desktop. You can instead click **'Start'** on the Windows menu. Then right click the **'My Computer'** menu item. Finally, select **'Properties'** from the menu prompted.
- 2 The **System Properties** dialog appears. Click the **Advanced** tag.

Windows NT/2000

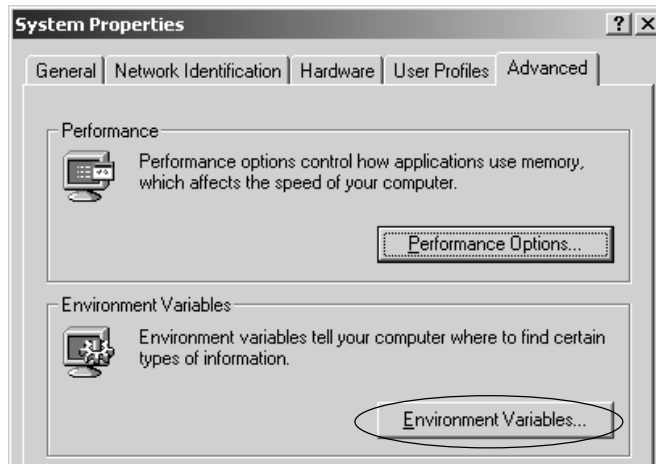


Windows XP

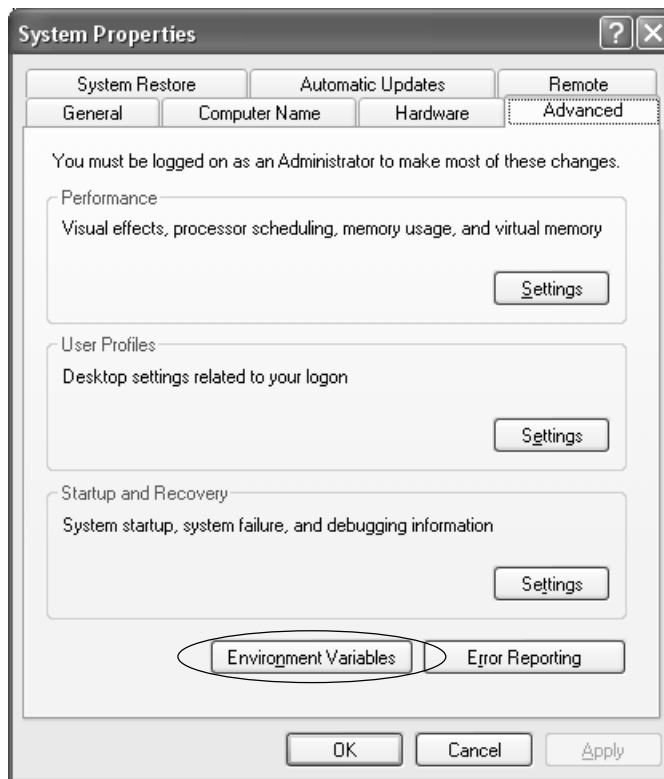


- 3 In the Advanced page, click '**Environment Variables...**'.

Windows NT/2000

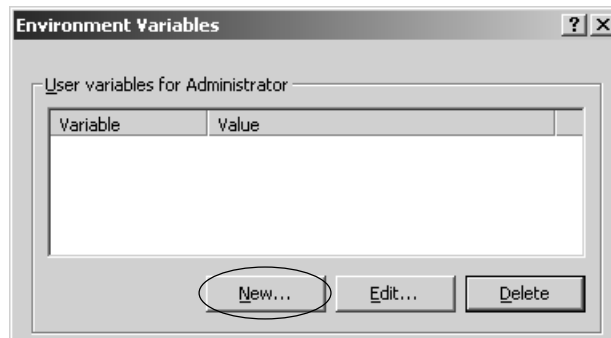


Windows XP

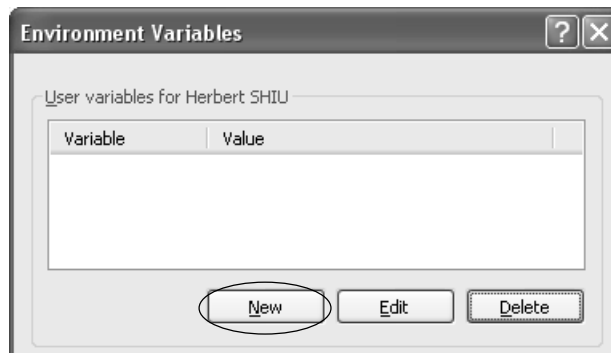


- 4 The **Environment Variables** dialog is shown. In the **User variables for Xxx** frame where Xxx is your login name, click '**New...**'.

Windows NT/2000

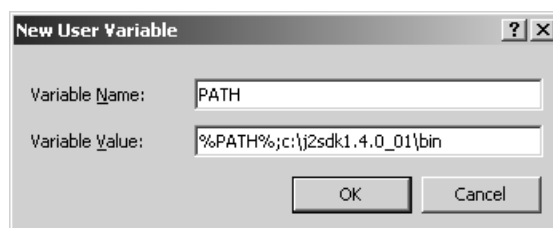


Windows XP

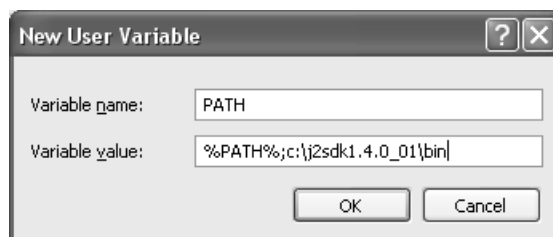


- 5 In the **New User Variable** dialog, enter 'PATH' in the 'Variable Name:' text box and '%PATH%;c:\j2sdk1.4.0_01\bin' in the 'Variable Value:' text box (the part 'C:\j2sdk1.4.0_01' should be the same as the directory of your Java SDK installation). That is:

Windows NT/2000



Windows XP

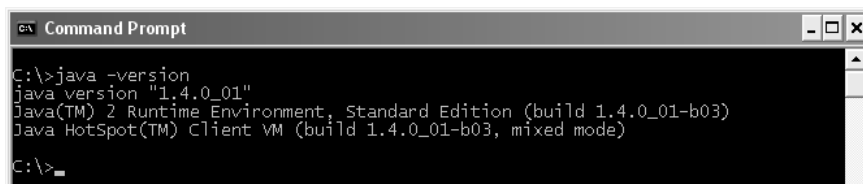


- 6 Click the **OK** button to close the **New User Variable** dialog.
- 7 Click the **OK** button of the **Environment Variables** dialog to close it.
- 8 Last, click the **OK** button of the **System Properties** dialog.

Checking the Java installation

The installation of the Java SDK is completed. To make sure your Java SDK is properly installed, please follow the steps below:

- 1 Click '**Start**' in the Windows menu. Select '**Programs**', '**Accessories**' and finally '**Command Prompt**' or '**MS-DOS Prompt**'.
- 2 In the '**Command Prompt**' or '**MS-DOS Prompt**', enter '`java -version`'. You should get the output as follows:



```
C:\>java -version
java version "1.4.0_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0_01-b03)
Java HotSpot(TM) Client VM (build 1.4.0_01-b03, mixed mode)
C:\>
```

- 3 If you get the following output,

`'java' is not recognized as an internal or external
command, operable program or batch file.`

or

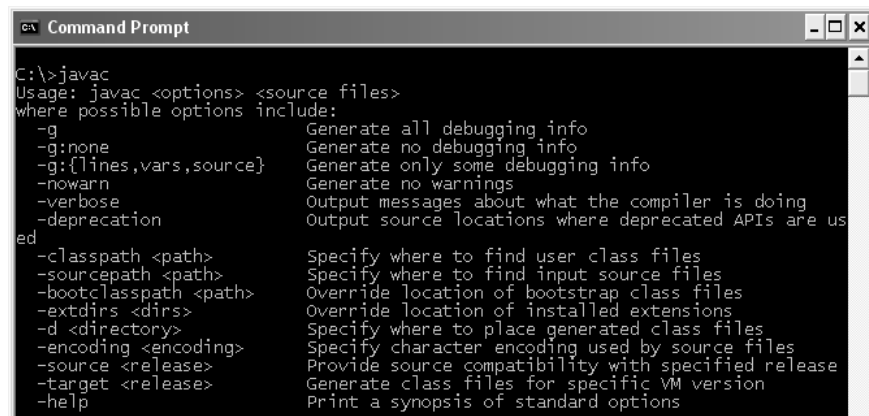
`Command not found`

or

`Bad command or file name`

your Java SDK has not been properly installed. Please check if your
PATH setting is correct. You may need to repeat the entire
installation procedure.

- 4 To ensure that the compiler is also properly set up, please enter 'javac' in the Command Prompt. You should get the following output:



```
C:\>javac
Usage: javac <options> <source files>
where possible options include:
-g          Generate all debugging info
-g:none     Generate no debugging info
-g:{lines,vars,source} Generate only some debugging info
-nowarn     Generate no warnings
-verbose    Output messages about what the compiler is doing
-deprecation Output source locations where deprecated APIs are used
-ed
-classpath <path> Specify where to find user class files
-sourcepath <path> Specify where to find input source files
-bootclasspath <path> Override location of bootstrap class files
-extdirs <dirs> Override location of installed extensions
-d <directory> Specify where to place generated class files
-encoding <encoding> Specify character encoding used by source files
-source <release> Provide source compatibility with specified release
-target <release> Generate class files for specific VM version
-help        Print a synopsis of standard options
```

If English is not the native language of your machine, you may get the output in the native language of your machine. Please don't panic. You can use the Java SDK exactly the same way as if English is the native language. When you compile your Java programs, the error messages may be in the native language of your machine as well.

You can now proceed to the next section to start experiencing the Java SDK.

Compiling and executing Java programs

You can now try the demonstration programs that come with the Java SDK.

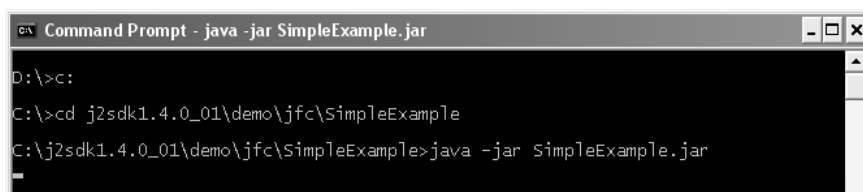
Executing a Java demonstration program

A standalone Java application is easier to start from the Command Prompt. If you are familiar with the Graphical User Interface (GUI) only, you should try to use the Command Prompt more so that you can perform the necessary operations using commands.

At the **Command Prompt**, enter

```
C:\>cd \j2sdk1.4.0_01\demo\jfc\SimpleExample
C:\>java -jar SimpleExample.jar
```

That is:



Then, the Java application is started. The following dialog appears on the screen:



Now you are sure that your SDK can be used for running Java applications.

Your first Java program

Here comes a historical moment in your programming life. You are going to prepare your first Java program, compile it and execute it.

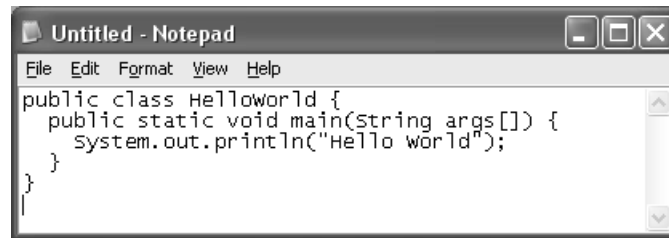
The contents of your first Java program are:

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello World");
    }
}
```

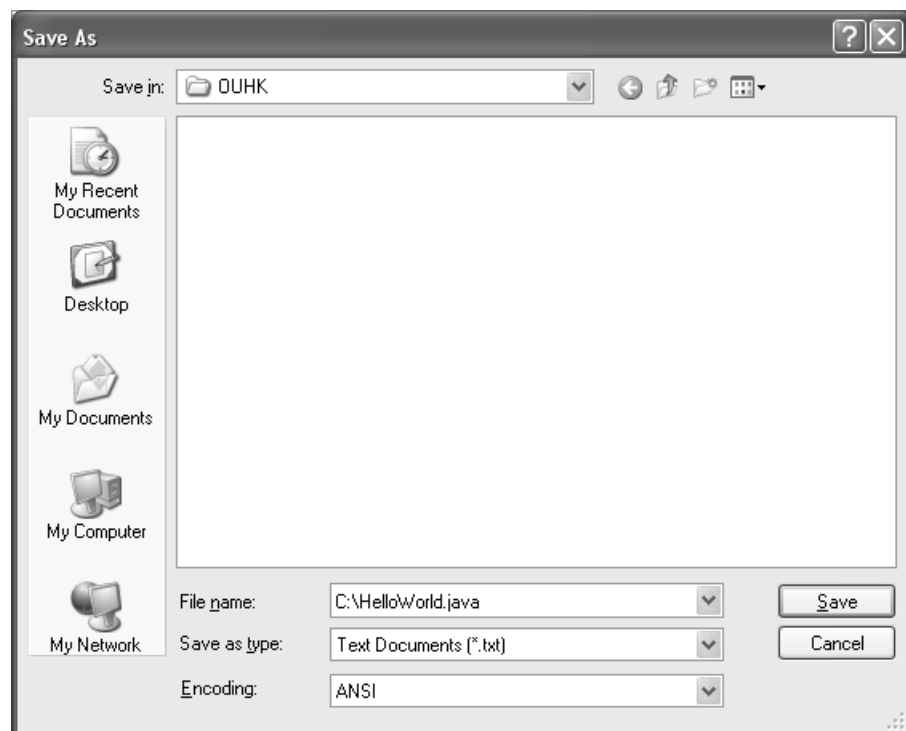
Please do not worry about the content of the above program. It is just for illustration for the time being. You will learn every detail of this program in later units.

Edit a Java program with Notepad

First of all, start the **Notepad** in your computer. If you have forgotten how to start **Notepad**, please refer to the previous section for details. Enter the program listed on the previous page into **Notepad**. Then, the content of the **Notepad** becomes:



Select '**File**' from the **Notepad** menu. Select the '**Save**' menu item. In the **Save As** dialog, enter '**C:\HelloWorld.java**' in the '**File name:**' text box:



Click '**Save**' to save the file to your hard disk.

Compile and execute your first program

Start a **Command Prompt** (if you haven't already done so). At the **Command Prompt**, enter:

```
c :  
cd \
```

to change the current directory to the root directory of your drive C.

You can now compile your first Java program with the Java compiler that comes with the Java SDK. Please enter:

```
javac HelloWorld.java
```

Java is a case-sensitive programming language. Therefore, you should use 'HelloWorld' instead of any other combinations of letter cases.

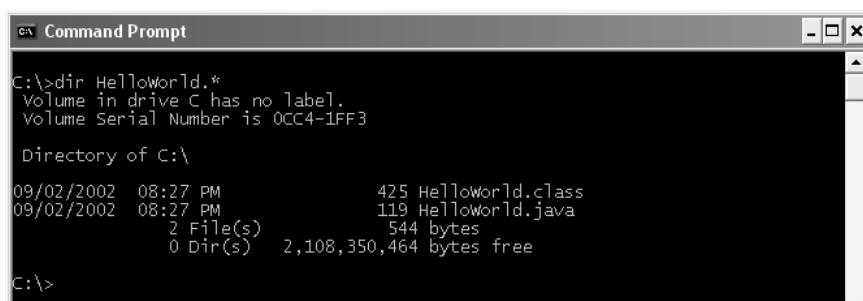
If you have entered the program exactly the same as the listing on the previous page, the above command will display nothing. The **Command Prompt** shows the prompt '>' after a few seconds.



```
Command Prompt
D:\>c:
C:\>cd \
C:\>javac HelloWorld.java
C:\>
```

Otherwise — *if you don't see the prompt or see a whole bunch of error messages* — the program you have entered is incorrect. Please verify and amend the contents in Notepad. Save the file and then try to compile the file again.

List the contents of the current directory. You will find a new file 'HelloWorld.class' is created if the compilation is successful:



```
Command Prompt
C:\>dir HelloWorld.*
Volume in drive C has no label.
Volume Serial Number is 0CC4-1FF3

Directory of C:\

09/02/2002  08:27 PM                425 HelloWorld.class
09/02/2002  08:27 PM                119 HelloWorld.java
               2 File(s)                544 bytes
               0 Dir(s)  2,108,350,464 bytes free

C:\>
```

The file HelloWorld.class is the resultant file after compilation. It contains Java bytecode that adheres to the JVM specification. Therefore, it is platform independent and you can transfer it to any other platform for execution without re-compilation.

To execute the program, enter:

```
java HelloWorld
```

and a message 'Hello World' will be displayed in the Command Prompt:



```
Command Prompt
C:\>java HelloWorld
Hello World
C:\>
```

If you get any message other than ‘Hello World’, you either entered wrong commands or the program contents are incorrect.

Common compilation and execution errors

There are two types of error that you might make when developing Java applications. They are compile-time error and run-time error.

Compile-time or compilation errors can be detected while you are compiling a Java source program. If you amend the Java source file and eliminate all the compile-time errors, it does not mean that your program will work properly. Run-time errors happen when you are *running* your compiled Java program.

The Java compiler will determine as many errors as it can during compilation. Some errors cannot be determined until the moment the problematic statement is executed. Then, you will get run-time errors.

A common compilation error is a typo. For example, if you prepared the following Java program

```

HelloWorld.java - Notepad
File Edit Format View Help
public class HelloWorld {
    public static void main(String args[]) {
        system.out.println('Hello world');
    }
}

```

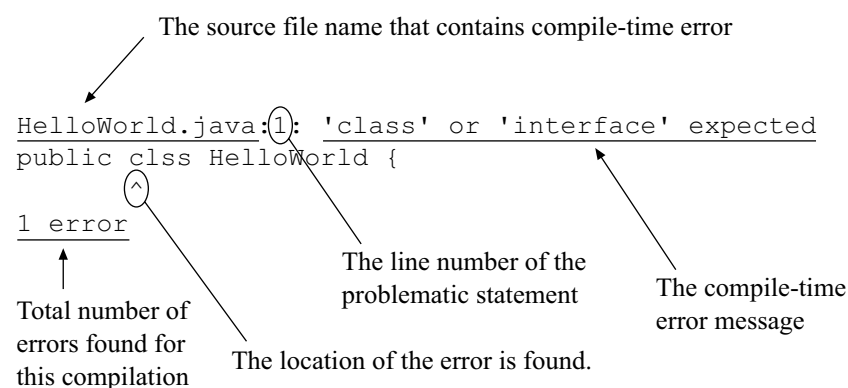
and compile it with ‘javac’, you’ll get the following compile-time error:

```

C:\>javac HelloWorld.java
HelloWorld.java:1: 'class' or 'interface' expected
public class HelloWorld {
    ^
1 error
C:\>

```

Let’s explore the generated error message:



By studying the compile-time error, you should find that the error occurs because the word 'class' is mistyped as 'clss'. By amending this typo in the program and re-saving it to the hard disk, you can compile the program again and will find that the error message no longer exists and the compiled code will run properly.

Compared with compile-time error, run-time error is much more difficult to determine. Compile-time errors are detected by the Java compiler, whereas run-time errors are detected by the Java Runtime Environment, which might occur any time the program is run. Therefore, you have to find out the situations in which run-time error occurs — and that is not a trivial task.

Let's consider the following Java program:

```

public class RuntimeError {
    public static void main(String args[]) {
        System.out.println(1 / 0);
    }
}

```

The program is syntactically valid and there is no compile-time error. When you compile and execute the program, you will get the following output:

```

C:\>javac RuntimeError.java
C:\>java RuntimeError
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at RuntimeError.main(RuntimeError.java:3)

```

In a few words, the program tries to display the result of the expression, one divided by zero (1/0). When the JVM executes the program and performs the division, it finds that the divisor is zero. Mathematically, a division of any number by zero is undefined. Therefore, the division operation in the program is undefined, and the JVM therefore halts the program execution and displays a run-time error message:

Exception in thread 'main' java.lang.ArithmeticException: / by zero
 at RuntimeError.main(RuntimeError.java:3)

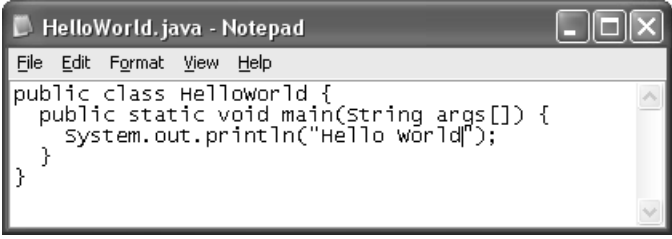
The source file name and the line number of the statement that generates the run-time error

The run-time error message

The methods of determining run-time errors are further discussed in later units.

Modify and re-execute your program

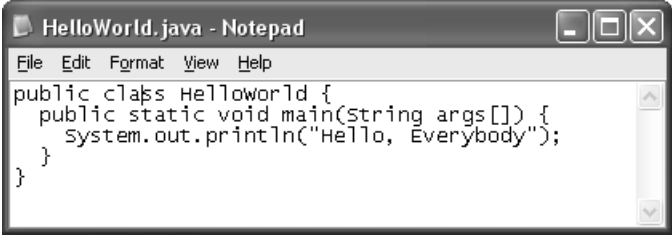
By studying the following program,



```
File Edit Format View Help
public class HelloWorld {
    public static void main(String args[]) {
        system.out.println("Hello world");
    }
}
```

it is obvious that the pair of double quotation marks encloses the message to be displayed at the Command Prompt. Therefore, by modifying the content of the message enclosed, the message to be displayed is changed accordingly.

For example, change the message from 'Hello World' to 'Hello, Everybody'. That is:



```
File Edit Format View Help
public class HelloWorld {
    public static void main(String args[]) {
        system.out.println("Hello, Everybody");
    }
}
```

Save the program and re-compile the program. You will get a different output:



```
C:\>javac HelloWorld.java
C:\>java HelloWorld
Hello, Everybody
C:\>
```

Self-test 1.4

Modify the HelloWorld.java source file so that the message to be displayed is:

Hello, <Your name>

For example:

Hello, Peter

Summary

Throughout the history of computers, researchers have invented different computer hardware and proposed various theories. Among them, the Analytical Engine by Charles Babbage and the von Neumann Architecture had profound effects on the development of computers, pioneering the stored program concept and the Fetch-Decode-Execute cycle respectively.

Any programmable device such as a calculator, a mobile phone, or a videocassette recorder can be considered a computer.

Hardware and software must cooperate to perform operations. Hardware is the physical devices that make up the computer; software is the data and programs to be executed by the computer hardware. Software can further be categorized into operating system and application software. Microsoft Windows a typical example of an operating system. Internet Explorer is an example of application software.

Programs need to be written in a programming language to instruct a computer to perform desired operations. Even though a lot of software is available for various purposes, it cannot fulfil the requirements of all users. It is therefore necessary to learn how to develop software. Programming languages can be classified into low- and high-level languages. The Java programming language is a high-level programming language and is object-oriented.

At the end of this unit, we discussed the steps to install the Java Software Development Kit (Java SDK) into your computer, so that you can use it for writing Java programs. We will discuss ways to write Java programs in the coming units.

References

A Brief History of Computer Technology,
<http://csep1.phy.ornl.gov/ov/node8.html>

A Little History of the World Wide Web, <http://www.w3.org/History.html>

Computer History Museum Home Page,
<http://www.computerhistory.org/>

Computers: History and Development
http://www.digitalcentury.com/encyclo/update/comp_hd.html

Java™ Technology: The Early Years,
<http://java.sun.com/features/1998/05/birthday.html>

Englander, I (year) *The Architecture of Computer Hardware and Systems Software, An Information Technology Approach*, 2nd edn, John Wiley & Sons, Inc.

The Source for Java™ Technology, <http://java.sun.com/>

Webopedia: Online Dictionary for Computer and Internet Terms,
<http://www.webopedia.com/>

Glossary

Algorithm — A pre-defined set of unambiguous steps for an operation.

Analogue-digital converter — The device that converts an analogue quantity into digital format so that the computer can process it.

Application Software — A software package that is targeted at performing particular operations such as word processing and sending email. It must match the operating system that runs on the machine, so that it can use the necessary resources and services.

Arithmetic and logic unit (ALU) — The device that performs arithmetic and logical operations in a computer.

ARPA/DARPA — The Advanced Research Projects Agency (ARPA) was formed in the Defense Department of the United States to investigate the applications of scientific knowledge in military areas. In 1971, it was renamed Defense Advanced Research Projects Agency (DARPA). In 1974, a project named 'Internetting' was carried out. The result of the project was the Internet.

ARPANET — The network designed by ARPA that was built in 1969. It is considered the predecessor of the Internet.

Assembly language — A low-level programming language that uses English-like words as vocabulary. Each instruction can be translated into a corresponding machine language instruction on a one-to-one basis.

Automation — A pre-set timely operation performed by computers. Computers can be set to perform a predefined set of operations based on the time, or they can perform pre-set operations if a condition or event occurs.

BASIC — Beginner's All-purpose Symbolic Instruction Code, developed in the mid-1960s. It is a high-level language that uses English-like vocabulary. There are many variants, such as Visual Basic developed by Microsoft.

Binary number system — A number system that represents numeric quantity with the digits 0 and 1 only.

Bit — The smallest data item a computer can store or manipulate. It can be either 0 or 1.

Byte — An ordered sequence of eight bits constitute a single byte.

Central Processing Unit (CPU) — The core part of a computer where operations take place. It consists of Arithmetic and Logic Unit (ALU), Control Unit and registers.

COBOL — A high-level programming language that targets the development of business applications, which has been around since 1954.

The name COBOL is an acronym for COmmon Business-Oriented Language.

Compilation — The process that reads the entire program written in high-level programming language and then generates the corresponding machine language to be executed by the machine. Compilation is carried out by compiler software.

Data — The raw materials to be presented to computer software for processing.

Digital-analogue converter — The output of computer software is in digital format. The digital-analogue converter converts the digital format data into the corresponding analogue quantities.

Electronic mail — Since the early stage of the Internet, electronic mail (also known as email) has been around, enabling the users to send textual messages to each other.

ENIAC (Electrical Numerical Integrator And Calculator) — The first general-purpose electronic digital computer built in 1945 at the University of Pennsylvania. It was built with 18,000 vacuum tubes and occupied 1,800 square feet of floor space.

File transfer protocol (FTP) — A standard mechanism for sharing files in the Internet. The access to the files can be open or password protected.

FORTTRAN — FORTRAN stands for FORMula TRANslator. It is the oldest high-level programming language, designed in the late 1950s. It is mainly used in developing scientific applications.

General-purpose computer — A generic computer system that performs any operations by loading different programs.

Hexadecimal number system — A number system that represents numeric quantity with 16 digits: 0 to 9 and A to F to represent 10 to 15 respectively.

High-level programming language — A programming language that uses human language as vocabulary; its structure is close to human languages. Computers cannot execute programs written in high-level programming languages, and either interpretation or compilation is needed to translate the instructions in the high-level programming languages into the machine languages of the machines that run them.

ICQ (I seek you) — A collection of communication methods including real-time textual or voice chatting and off-line email messages. The users of ICQ have formed a community, each of which is identified by a unique ICQ number.

Imperative programming language — An imperative programming language supports performing operations sequentially on data. It is also known as procedural language.

Information — The results of computer processing based on the data provided.

Instruction Pointer or Program Counter — A special register in a computer that stores the address of the next instruction to be executed.

Integrated Circuit (IC) — Many transistors are put on a chip to form an integrated circuit. Computers built with ICs can be more compact and consume less electrical power and generate less heat than transistors do. Because of the advances in technology, Large Scale Integration (LSI), Very Large Scale Integration (VLSI) and Ultra Large Scale Integration (ULSI) have become possible.

The Internet — The network that evolved from a funded research project named 'Internetting' by DARPA in 1974. It connects many computers worldwide.

Interpretation — The process that reads a program written in a high-level programming language — one instruction at a time — and translates it and then executes it. The software that performs interpretation is interpreter software.

Java applet — A specially written Java program that must be executed in a Web browser such as Internet Explorer and Netscape.

Java application — Standalone application software that is written in the Java programming language.

Java — A high-level object-oriented programming language, formerly called Oak, which was created for handheld devices and set-top boxes. In 1995, it was renamed Java and targeted at the Internet.

JavaScript — An interpretive programming language that can be embedded in Web pages to be executed by a Web browser. It enables interactive Web pages. It is not related to Java.

Low-level programming language — Machine language and assembly language. They are machine dependent.

Machine language — The native language that can be understood by a machine. It is composed of a sequence of 0's and 1's. It is considered to be the first generation programming language.

Memory/Primary storage — The storage that stores data, programs and the immediate operation results. Primary storage is volatile — data stored in it are lost if power is lost.

Moore's Law — An observation made by Gordon Moore, a co-founder of Intel, who claimed that computer processing power doubles every 18 months.

Object code — Programs written in high-level programming language can be compiled into machine language for execution. The result of such

compilation is known as object code, which is usually in the machine language of the machine that executes the program.

Object-oriented programming — The programming paradigm that supports encapsulating states and operations as objects and message passing among objects.

Octal number system — A number system that represents numeric quantity with 8 digits, from 0 to 7.

Office automation — The use of computer software to perform various office operations, such as word processing, bookkeeping and so on.

Operating System — An essential software package to be loaded into a computer when it starts. It manages and allocates all computer resources such as memory and provides necessary services such as authentication to other application software.

Peripheral — A non-essential device connected to a computer system, which either obtains input data from the external environment to the computer system or generates the outputs according to the operation results.

Perl — A high-level programming language. Perl stands for Practical Extraction and Report Language. It is an interpretive programming language that was designed for processing text.

Personal computer (PC) — A generic computer system targeted for use in the home, office and school. IBM first introduced it in 1981.

Program counter — See instruction pointer.

Programming language — A well-defined language, including vocabulary and a set of grammatical rules, for instructing a computer to perform desired tasks.

Random access memory (RAM) — Stores data and programs when there is electrical power supply. A computer loads the data and programs from the secondary storage. RAM is used for storing data and the program to be executed and the immediate operation results (see memory/primary storage).

Read only memory (ROM) — The storage device that can maintain data and programs with or without electrical power supply. The contents are usually the utility programs that control various hardware devices and load the operation system to the memory when the computer starts.

Secondary storage — The storage devices that can maintain the data and programs even without electrical power. Examples are hard disk, CD-ROM and so on. A computer cannot process data and programs that are stored in secondary storage — they must be loaded into primary storage for processing.

Software — The data and programs presented to a computer for execution and processing. You can physically touch the media that store the data and programs but not the stored media. Therefore, the stored media are referred to as software.

Source code — A program written in any high-level programming language.

Special-purpose computer — A computer system designed for a special or dedicated purpose. The programs to be executed are usually stored in the primary storage, and they can survive a cut in electrical power supply.

Structured query language — A standard query language used mainly for retrieving data from a database system. It is considered to be a fourth-generation language.

Telnet — It emulates a terminal to a machine through a network so that more than one user can share a single machine and that machine is accessible from anywhere across the network.

Transistor — Transistors replaced vacuum tubes for building computers. Computers built with transistors are called second-generation computers. They were smaller, consumed less electrical power and generated less heat.

Vacuum tube — The building block of first-generation computer systems. Compared with the upcoming enabling technologies, it is bulky, consumes much more electrical power, generates much more heat, and is unreliable.

Virtual reality — An environment artificially created by a computer to give the user an illusion that the user is in a real environment. Special equipment, such as gloves and goggles, are usually required.

Visual Display Unit (VDU) — An output device that can generate visual output, which usually refers to a computer monitor. There are two main types of monitor — the Cathode-ray tube (CRT) and Liquid crystal display (LCD). VDU can also be special goggles used for Virtual Reality.

von Neumann architecture — The computer architecture proposed by John von Neumann. The architecture outlines the basic components a general computer and their interrelationship.

World Wide Web (WWW) — The system that enables distribution of documents and resources, such as multimedia files on the Internet. The documents are written in hypertext mark-up language (HTML). Each document is assigned a unique identifier so that documents can refer to one another.

Suggested answers to self-test questions

Self-test 1.1

- 1 The steps you take when you check out at the supermarket cashier could be:
 - a Check if all desired items are collected. If some desired items are still missing, repeat step a.
 - b Present the collected items to the cashier.
 - c Wait for the total price of all items.
 - d Prepare the amount of money that is equal to or greater than the total price. Present it to the cashier.
 - e Check if the change is correct. Otherwise, negotiate with the cashier and repeat step e.
 - f Finish, and take the items home.

Note: if your answer is not the same as the one above, that doesn't make your answer wrong. Remember, there's no one way to solve a problem.

2

Cases	Number of possibilities	Time required
A word can be found in a dictionary	$= 200000000$	$= 200000000 / 1000000$ $= 200$ seconds
An arbitrary combination of 8 lowercase letters	$= 26^8$ $= 2.08827 \times 10^{11}$	$= 2.08827 \times 10^{11} / 1000000$ $= 58$ hours
An arbitrary combination of 8 lowercase or uppercase letters	$= 52^8$ $= 5.346 \times 10^{13}$	$= 5.346 \times 10^{13} / 1000000$ $= 5.346 \times 10^7$ seconds $= 618.7$ days
An arbitrary combination of 8 lower case letters, uppercase letters and digits	$= 62^8$ $= 2.183 \times 10^{14}$	$= 2.183 \times 10^{14} / 1000000$ $= 2.183 \times 10^8$ seconds $= 6.92$ years

You can see that choosing a word that can be found in a dictionary is of no use, as no password is used. Therefore, a 'good' password should contain at least eight characters and be composed of arbitrary combinations of lowercase letters, uppercase letters and digits.

Self-test 1.2

- 1 Converting the numbers from other number systems to the decimal number system:

$$\begin{aligned} 10101101_2 &= 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 128 + 0 + 32 + 0 + 8 + 4 + 0 + 1 \\ &= 173 \end{aligned}$$

$$\begin{aligned} 6457_8 &= 6 \times 8^3 + 4 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 \\ &= 3072 + 256 + 40 + 7 \\ &= 3375 \end{aligned}$$

$$\begin{aligned} ABCD_{16} &= 10 \times 16^3 + 11 \times 16^2 + 12 \times 16^1 + 13 \times 16^0 \\ &= 40960 + 2816 + 192 + 13 \\ &= 43981 \end{aligned}$$

- 2 The octal digits 2 and 3 are represented as binary numbers 010 and 011 respectively. Therefore, the binary number for 23_8 is 010011 or simply 10011.

The hexadecimal digits A, 1, B and 2 are 1010, 0001, 1011 and 0010 respectively. Therefore, the equivalent binary number is obtained by concatenating them. The result is 1010000110110010.

$$\begin{aligned} 3 \quad 11110000 &= 11 \quad 110 \quad 000 &= 360_8 \\ 11110000 &= 1111 \quad 0000 &= F0_{16} \\ \\ 100010001 &= 100 \quad 010 \quad 001 &= 421_8 \\ 100010001 &= 1 \quad 0001 \quad 0001 &= 111_{16} \end{aligned}$$

Self-test 1.3

Based on the assumption that each word, on average, takes 6 bytes (5 letters and 1 space) to store, the average number of words on a page is 250, which needs 1500 bytes for storage. Therefore, the number of pages that can be stored is:

$$\begin{aligned} &= 650 \times 1024 \times 1024 \div 1500 \\ &= 454382.933 \end{aligned}$$

Therefore, a CD-ROM can roughly store 454,383 pages of documents.

Self-test 1.4

To modify the message to be printed, it is just necessary to modify the message enclosed with double quotation marks. The program should be modified as:

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello, Peter");
    }
}
```

Feedback on activities

Activity 1.1

To construct the table for guessing ages, allocate the numbers from 1 to 63 to the table in the following way:

- 1 Convert each number into its equivalent binary numbers with 6 digits. If the first binary digit is 1, place the number in column 1. Repeat the process similarly for column 2 and so on. For example, the binary number of 27 is 011011. The number 27 is therefore placed in the columns 2, 3, 5 and 6.
- 2 Sort the numbers in the columns in ascending order.

As the table is constructed in this way, if an age can be found in a particular column, the corresponding binary digit in the six-digit binary number is 1. Otherwise, the corresponding digit is 0. Therefore, the age can be found in columns 2, 3, 5 and 6, which means that the number is equivalent to 011011. In other words,

$$\begin{aligned} &= 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 0 + 16 + 8 + 0 + 2 + 1 \\ &= 27 \end{aligned}$$

Since the numbers at the top left corners of the tables are 32 (2^5), 16 (2^4), 8 (2^3), 4 (2^2), 2 (2^1) and 1 (2^0) respectively, the age to be determined is simply the sum of the numbers at the top left corners of the columns that contain the age.