

Revision: What's programming

- A computers contains a CPU, some memory (RAM), hard disk(s), and other peripheral devices.
- The CPU is responsible for controlling the rest of the computer.
- CPU achieves this task by following exactly some instructions, which resides in the memory.
- Programmers works to create instructions.
- The act to create instructions is called programming, and is the core subject of this course.

0911B L02-1

Instructions of a computer

- **Instructions are simple:** to make it cheap to manufacture CPU.
- CPU contains a small amount of space (registers) for temporary storage of intermediate results.
- CPU instructions can:
 - Read or write something in the memory. (e.g. *Load the 102,796-th memory word to the third register.*)
 - Doing arithmetics. (e.g. *Add up the two numbers in the second and third register and put it into the fourth.*)
 - Input from or output to a peripheral device. (e.g. *Read the letter pressed on the keyboard to the third register.*)
 - Choose the next instruction to run. (e.g. *If the number in the second register is smaller than that in the third, go back 15 instructions.*)

0911B L02-2

Computer programs

- When people use computers, they do run individual instructions. Instead, they run programs.
- A program is a sequence of instructions that are arranged carefully by programmers to do a specific task.
- Users select a program to run by various ways. For example, in Windows 95/98 you may:
 - Click the mouse button on an icon on the desktop;
 - Use the Start menu to choose a program;
 - Use the Start menu to choose "Run", and type in a program name;
 - Type the program name in the DOS window.

0911B L02-3

People don't want to write instructions

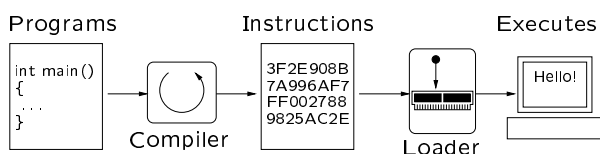
- CPU instructions are in **machine language**, designed to be read by machines, not by people. People find it too troublesome to work with them.
e.g. People want to say "Print the word 'Hello' to the monitor", rather than "put the words 'Hello' in memory, call the routine to print the words, passing it that memory location".
- Different computers understand different kinds of instructions: programs need to be rewritten for each type of computers.
e.g. The kind of instructions for Pentium, Palm Pilot and Mac are all different. If your program need to run in each of them, you need to write it 3 times.
- People do not want to specify exactly the instructions.
e.g. People want their programs to be rearranged automatically to best utilize the CPU registers.

0911B L02-4

Programming process

- Programs are written in a **high level language**, not machine language.
- By design, people can easily work with high level languages.
- A program, called **compiler**, is used to convert the program into the correct machine language.
- The compiler automatically tries to speed up the program by re-arranging the instructions.

"Normal" programming process:

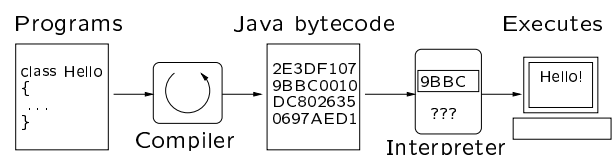


0911B L02-5

The Java Programming process

- Java is more than a language: it also defines the machine that runs Java instructions (Java Virtual Machine).
- In reality no such machine exist. So instead of a program loader, it requires a Java interpreter to run a Java program.
- This arrangement allows a program to be compiled to "Java bytecode" only once, and run on every type of machines.

Java programming process:



0911B L02-6

Java platform

- A Java platform provides all the tools needed for the writing and running programs in Java.
- We will use the standard Java platform provided by Sun Microsystems free of charge. The Java compiler and bytecode interpreter are called `javac` and `java` respectively.
- There are commercial alternatives: Inprise (previously known as Borland) provides JBuilder, and IBM provides VisualAge Java. They ask people to pay for better performance and better programming environment.
- The standard Java platform can be downloaded from its web page: <http://java.sun.com/products/jdk/1.2/>.
- We will demonstrate the installation of Java in the first workshop.

0911B L02-7

A Java Program

```
1 /* Purpose:
2     Prints out "Hello World".
3 */
4 public class HelloWorld {
5     public static void main(String[] args) {
6         System.out.println("Hello World!"); // For illustration
7     }
8 }
```

- This is a **Java Program**, to be typed into the computer using an editor and saved as a file called `HelloWorld.java`.
- After that you create a DOS prompt, and compile the program with `javac HelloWorld.java`. This gives you the byte-code of the program in `HelloWorld.class`.
- Then you can run the program by typing `java HelloWorld`.
- The program simply prints out "Hello World" and exits.

0911B L02-8

Explanation of the program

- Line 1–3 contains a **multi-line comment**. The compiler ignores everything between `/*` and `*/`. They are intended for people. You should have one comment at the beginning of each program to tell others what it does.
- Line 4, 5, 7 and 8 are statements that helps you organize the program. You will need such line in each of your program, and except the word "HelloWorld", everything is exactly the same.
- Line 6 is the core of the program. It basically says: *Tell the system output device (i.e. monitor) to print the words "Hello World!" and then end the line.*
- Note that you can more or less guess the meaning.
- Line 6 contains a **single-line comment**. It starts with two slashes and continue until the end of the line. Again the compiler ignore the comment, it is for people to understand the program easily.

0911B L02-9

Names

The program has a name, `HelloWorld`. The name must match the filename of the program, or it will not compile right.

There are some rules governing names in Java. In the remainder of the course we will learn more things in Java that have a name (identifier). All of them uses the same rule.

- Identifiers can contain only letters, digits and the underscore (`_`) character.
- The first thing in an identifier must be a letter.
- There are some words that cannot be an identifier in Java. (e.g. *It would be rather confusing if the name of a program is "class".*) These words are said to be "reserved words".

Examples identifiers: `HelloWorld`, `double_side`, `pi2`.
Counter-examples: `Hello World`, `2side`, `pi-2`, `class`.

0911B L02-10

List of reserved words

These words have special meanings in Java and cannot be used as an identifier.

abstract	boolean	break	byte	case
catch	char	class	constant	continue
default	do	double	else	extends
false	final	finally	float	for
goto	if	implements	import	instanceof
int	interface	long	native	new
null	package	private	protected	public
return	short	static	super	switch
synchronized	this	throw	throws	transient
true	try	void	volatile	while

0911B L02-11

The core of the program

The core of the program, line 6, is a **simple statement**:

```
6     System.out.println("Hello World!"); // For illustration
```

- Simple statements are always terminated by a semicolon (`;`). The compiler will continue to read lines until its sees one.
- They do exactly one thing once.
- In this case, it causes the words "Hello World!" to be printed.
- You can output any simple thing ("of primitive type") in Java to the screen in this way.
- If you do not want to end the line, use `System.out.print` instead.
- There are a pair of double quotes (`"`) enclosing some words. It is called a Java string, and will be covered in the next lecture.

0911B L02-12