Welcome to Object-Oriented Programming Using Java

Lecture 1

- Covers
  - Overview and general information about the subject
    - Aims, design, teaching approach, etc.
    - Resources
    - Requirements for passing OOJ/OJA
  - Introduction to computers

- Reading: Savitch 1.1

- Subject Information

Subject aims

- To understand how computers work
- To develop techniques for solving problems using a computer
- To develop skills in algorithm development and object-oriented programming using the Java language
- To gain experience in engineering reliable programs

… in particular

- OOJ/OJA and IPJ/IJA have been designed to form the core of Java programming
- They provide a solid foundation for Java application development

Teaching philosophy

- Learning is an active process
- The role of the teaching staff is to assist students to learn
- Students must be self-driven and self-managed
- Learning in groups is a positive experience
- Problem solving and programming are best learnt by practice

References

- Textbook
  - Walter Savitch, *Java: An Introduction to Computer Science & Programming (Third Edition)*, Pearson, 2004
- Recommended References
  - Cay Horstmann, *Big Java*, John Wiley & Sons, Inc., 2002
  - Walter Savitch, *Absolute Java*, Pearson, 2004
  - Daniel Liang, *Introduction to Java Programming (Fourth Edition)*, Prentice Hall, 2002
  - Harley Hahn, *Student Guide to Unix*, McGraw-Hill, 1996
  - Steve Oualline, Vi IMproved – Vim, New Riders, 2001 (http://www.newriders.com/books/opl/ebooks/0735710015.html)

What is expected from you?

- An interest in the material
- Participation in labs and lectures
- Be aware of the learning environment you share with other students
  - No disruptive behaviour

Overview of OOJ/OJA

- Introduction to Computer Systems,
  - Unix Environment and Java Programming
- Primitive Data Types
- Control Structures
- Classes and Objects
- Arrays
- Applets (Graphics, Events and Animations)

- Computer Systems and Structure

What is a computer?

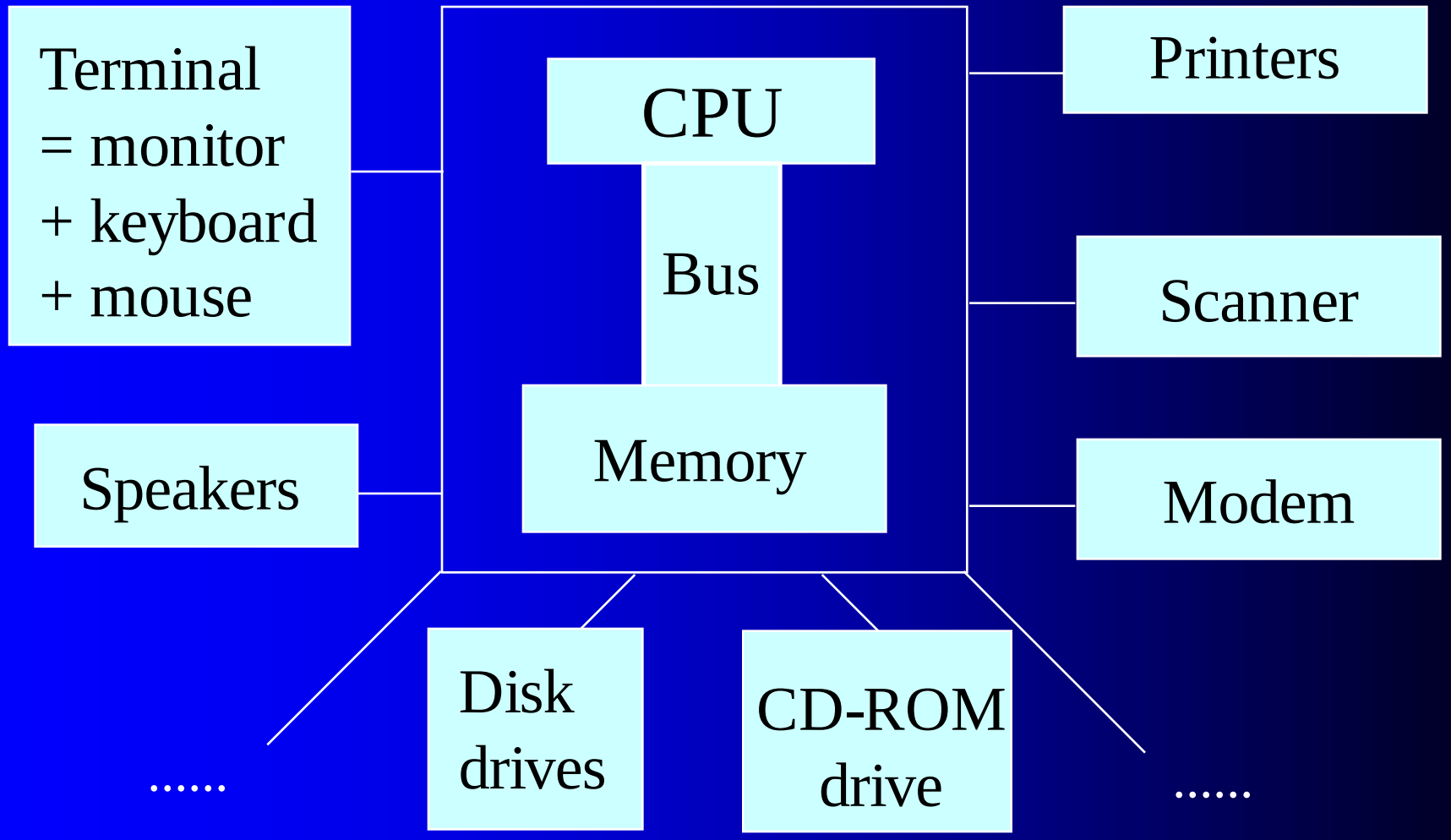
- A computer is a device for storing and processing information
- When connected to networks, a computer becomes a versatile communication tool
- When connected to other devices, a computer can act as a sophisticated device-controller
- Java provides facilities to write programs to create all those functions

Hardware and software

- A computer consists of hardware and software
- Hardware consists of the physical components that make up the computer
- Software consists of programs that control the hardware to do useful work
- Programming (or coding) is the process of writing software

Examples of hardware

Typical components in a PC



Main hardware components

- CPU - Central Processing Unit
  - Executes program instructions
- RAM - Internal memory
  - Stores programs that are currently being run and data associated with those programs
  - Volatile
- I/O (Input/Output) devices
  - Allow information to be input to the computer (e.g. keyboard, mouse, scanner)
  - Allow information to be output by the computer (e.g. monitor, printer)

Main hardware components

- Secondary storage (or external memory)
  - Disk, tape, CD-ROM
  - Larger, slower, cheaper than internal memory
  - For storage of programs and data not currently in use
- Bus
  - Transmits data and instructions between the CPU and memory
  - n address bits means we can address 2^n memory locations labelled 0 to $2^n - 1$

Terminology and functionality

- Files and directories
  - Ways of storing and organising data and programs on secondary storage
  - Covered in detail in laboratory classes
- Peripheral devices
  - Collective name for secondary storage and I/O devices

Data vs information

- What is data?
  - Series of (numeric or character) values in some representation
- What is information?
  - Data plus an associated meaning or interpretation

Binary representation

- Inside the computer, data and instructions are represented as bit patterns
- Bit
  - A single 0 or 1*
- Byte
  - 8 bits

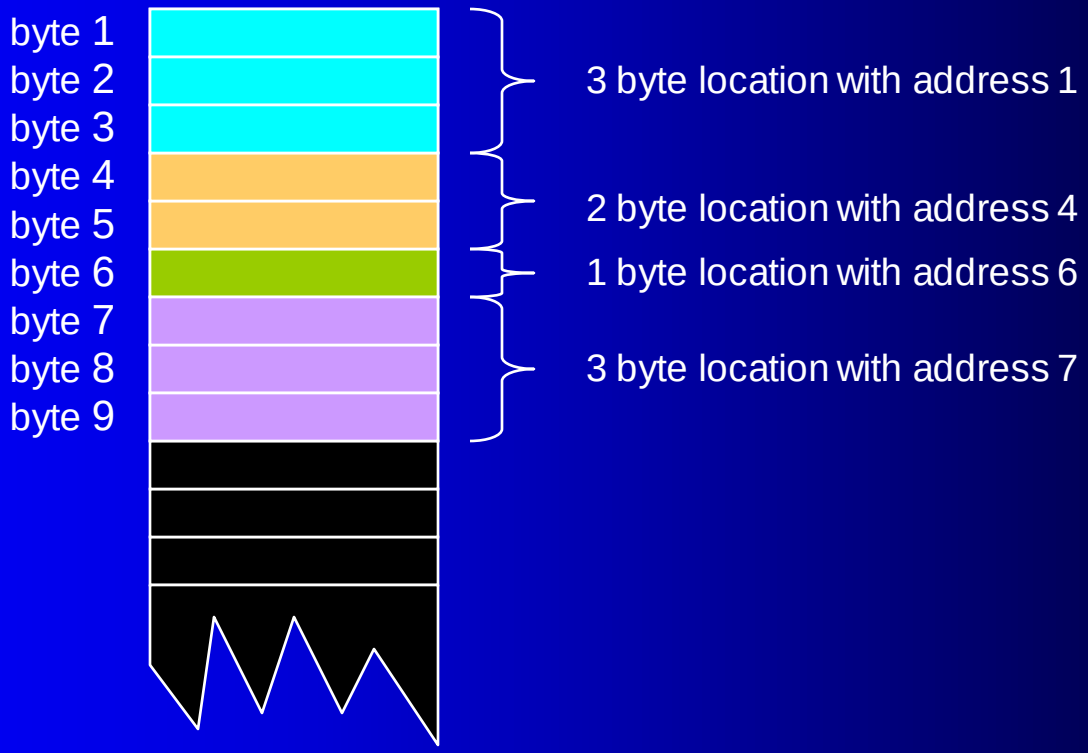
** Why 0's and 1's?*

Memory organisation

- A computer's memory is organised as a sequence of bytes
- A byte's location is called its address

Memory organisation

- Memory



Programming languages

- A program in the computer is a long sequence of bits
- But programs can be written in
  - Machine languages
  - Assembler languages (use symbolic names for instructions)
  - High-level languages (such as Fortran, Cobol, Pascal, C, C++, Java, etc.)
- Each programming language has a defined set of rules that govern the structure of its program

Machine code

```
01000   10000000000000000000010100001
01001   10110000000000000000010100010
01010   10001101000000000000000001111
01011   10000010000000000000010100011
01100   10000000000000000000010100010
01101   10110100000000000000010100011
01110   10001100000000000000000010000
01111   10000001000000000000000000000
10000   10000010000000000000010100010
```

- A group of bytes (i.e. a sequence of bits) can represent many different things: a *number* *, a *character*, a *string*, a *picture*, a *sound track*, a *video*, or a *program instruction*

** a real number (one with a fractional part) is represented by two sequences of bits, one for the mantissa and one for the exponent*

- e.g. the bit sequence 01000001can be interpreted as
  - Binary number
    - `01000001`
  - Hexidecimal
    - `41`
  - Character
    - `'A'`
  - Address
    - `01000001`
  - Decimal
    - `65`
  - Assembler instruction
    - `LDI`

In assembler

```
10  LDA 241
11  SUB 242
12  JZE 17
13  STO 243
14  LDA 242
15  DIV 243
16  JMP 20
17  LDI 0
20  STO 244
```

The assembly code clearly shows the kinds of *primitive* instructions actually executed by the CPU

In Fortran IV

```
      IF (X.EQ.Y) GO TO 10
      Z = Y / (X - Y)
      GO TO 20
10    Z = 0
20    {next statement}
```

In Java

```
if (x == y)
        z = 0;
else
        z = y / (x - y);
{next statement}
```

Algorithms

- Consider the algorithm (an unambiguous sequence of instructions to perform some task)

```
IF X equals Y THEN
        Assign zero to Z
ELSE
        Assign Y / (X - Y) to Z
ENDIF
{next statement}
```

Pseudocode

Types of software

- Computer software is usually divided into broad categories
- System software
- Application software

System software

- Supports basic operations of the computer such as managing resources, allowing transfer of information to and from the computer
- Examples
  - Operating systems (e.g. Unix, Windows)
  - Communication software (to connect to other computers and the Internet)
  - Compilers (translate high-level programs into executable code)

Application software

- Helps users to carry out specific tasks, e.g. creating a document
- Examples
  - Word processors
  - Spreadsheets
  - Other programs, for example programs to support operations of organisations

Next lecture

- Operating systems
- The Unix operating system
- Compiling and running Java programs