Aims of Module 1:

• Become familiar with using the ECLIPSE package to develop basic Java programs.
• Understand some of the basic concepts behind object-oriented programming.
• Be able to write a simple Java program.
• Understand the basic elements of the Java programming language: data types & variables, functions and algorithm control.
What’s This Course All About?

Electron

Atom

Nucleus

Proton

Nucleus

Object-Oriented Programming

contains

contains

contains

contains

attracts

using
Why Do We Write Computer Programs?

• Perform calculations.
• Read, manipulate and store data.
• Simulate some aspect of real-life ("abstraction").
• [Play games]

• All programs will therefore contain:
  – **Data**: describe/store the state of a system.
  – **Functions**: manipulate/change the state of a system.
Programming Languages

- It’s useful to have a very simple model of a computer in mind:
  - Ultimately a CPU only understands very simple instructions such as:
    - “read contents of memory location 0x15a7”
    - “increment register 0xa74f by 1”
    - … etc.
Low-Level and High-Level Languages

• Writing a program using such “low-level” instructions would be impossibly tedious.
• Some 50 years ago, “high-level” languages were developed.
• These languages contain logical and arithmetic instructions that are much easier to read, write and understand.
• The disadvantage is that they must be “compiled” or translated into the simple instructions that a computer can understand, before they can be run.
• The Java language appeared on the scene in 1995, as a high-level object-oriented (“OO”) language. It was designed to be a highly portable language “Write Once, Run Anywhere”. This often makes it the language of choice for web applications, for example.
Bits and Bytes and All That

Let’s recall some basics:

• A bit can take the value 0 or 1.
• A byte is comprised of 8 bits.
• Longer “words” consist of 32 or 64 bits.
• Integers are represented in binary format, for example:
  \[00101101\] = 45

• The representation of real numbers (floating point numbers) is slightly more complicated.
• All high-level computing languages have representations of commonly used entities: logicals, numbers and characters:
  – boolean (true/false), integer, float, character
Some Object-Oriented Nomenclature

• OO languages infinitely extend the range of possible types.
  – Not just boolean, integer, float, ...
  – But vector, matrix, particle, electron, atom, etc.
• We can write much more natural looking code using these programmer-defined types.
• These new types have their own class definitions. This is a bit like the “mold” or “archetypal” representation of an ideal object.
• A program creates or instantiates real objects according to their class definition. Different objects of the same type can have different values for their properties.
But What Is An Object?

- Logically, an object can be thought of as a kind of “bundle” of data and functions (or “methods”) that act on the data.

- The tight “coupling” between the data and functions comprising an object is a very powerful way of organising and managing a large programming task.

- A program advances by objects communicating with one another through the passing of “messages”.

![Diagram of object communication through message passing]
Some Concepts of Object-Oriented Programming (1)

Encapsulation

- Closely related to “information-hiding”.
- In order for an application programmer to be able to use objects of type \( X \), he need only be familiar with the \textit{interface} of that type.
- He does not need to be concerned with the internal workings of the object. In this sense, the objects are “black-boxes”.

- Makes code much more maintainable - as long the interfaces are stable.

Do not need to know the details of how the energy is stored and/or calculated on the fly.
Inheritance

- An apple is a type of fruit. An electron is a type of lepton.
- Object-oriented languages provide a formal structure called “inheritance” for expressing these relationships.

Exploiting inheritance and other relationships between objects is critical to good OO program design.
Some Concepts of Object-Oriented Programming (3)

Polymorphism

• “One entity has many forms”.
• Polymorphism itself can mean many things!
• One example: 2 objects may respond in different ways to the same message from another object.
  – Suppose a program contains a “shape” object.
  – The “shape” object is asked what its area is.
  – If the “shape” object is in fact a “rectangle” then it will return “length × height”.
  – If the “shape” object is in fact a “triangle” then it will return “0.5 × based × height”.

For Today …

- Essentially you can forget all about objects until Module 2.
- Today we are concerned with basic Java syntax and how to compile and run simple programs.

Start on the Module 1 Exercises
Please ask Questions!