Unit 2: Programming languages

Object oriented programming (OOP)

Key terms

Term	Definition
OOP	Programming based on collections of objects that interact with each other.
Object	An object is an instance of a class. An object can be a data structure, a variable or a function and will have an allocated memory location.
Class	A class is an entity that determines how an object will behave and what the object will contain. It is a template or a set of instructions to build a specific type of object.
Methods	A method is an action or behaviour that an object is able to perform.
Encapsulation	The process of wrapping data and the code that operates on it into a single entity. The variables and methods are wrapped up inside the class.
Abstraction	The process of hiding non-essential features and showing the essential features.
Inheritance	A new (derived) sub-class inherits the states and behaviours of the existing (base) super class.
Polymorphism	The concept that allows actions to act differently based on the object performing the action.

Some advantages of OOP

- Conceptual: Program objects are modelled on real world objects.
- Modularity: The source code for an object can be written and maintained independently of the source code for other objects.
- Code re-use: If an object already exists it can used in other programs. This allows specialists to implement/test/debug complex, task-specific objects.
- Debugging: If a particular object turns out to be problematic, it can be replaced with an object that functions correctly.

Objects and Classes

Real world objects have certain states. For example, a glass bottle can be full, empty, or somewhere in between. It can have a colour, weight, diameter and circumference.

A software object modelled after a glass bottle could have Boolean values for full or empty and a 'real' value for interim states. The glass bottle object could also have variables to represent colour, size, etc.

The glass bottle could then be used as a base class, or super class for other, more specialised bottles. For example, Energy drink bottle could become a sub-class of Glass bottle, that only allows for certain colours and sizes.

Class Variables and Methods

Class variables hold attributes shared in common among all objects of a class. Class methods act on attributes of the whole class through class variables. Class methods cannot access instance variables.

Instance variables and methods are object specific.

Greenfoot

Course specification requirement to "Design, write, test and refine Java programs within the Greenfoot environment, using these skills:"

Create new and extend existing classes
Create new and edit existing objects
Create new and edit existing worlds

Write and invoke methods
Change existing methods
Create new and edit existing properties (including public, private, static, etc.)

Add and remove objects from worlds
Use actors
Move objects around a world

Keyboard input
Add and play sounds

Implement and use parameter passing (by value and by reference)
Access one object from another

Implement object collision detection
Implement random number generation
Use the concept of inheritance and encapsulation.

Greenfoot

A Java interactive development environment that allows the development of two-dimensional graphical applications, like simulations and interactive games.