OBJECTS AND CLASSES

CITS1001
Concepts for this Lecture

- object; class;
- method; parameter; signature;
- data type;
- multiple instances; state;
- method calling; source code; method result

- Reading: Objects First with Java, Chapter 1
Classes and Objects

A *class* is a group of *objects* that have similar characteristics and exhibit similar behaviour.

An *object* is a specific *instance* of a *class*.

Objects – represent ‘things’ from the real world, or from some problem domain
Example: the red car down there in the car park

Classes – represent all objects of a kind
Example: car, lecturer, student
Example

- The set of all students forms the class Student
- Each individual student is an object of the class Student
- John Smith and Janice Lee are instances of the class Student
Example

- The set of all dogs forms the class Dog
- Each individual dog is an object of the class Dog
- Spot, Rover and Rex are all instances of the class Dog
Why do we use classes?

• To reduce complexity
• We can often deal with a specific object based purely on knowledge of its *class*, rather than that particular *instance*
• For example, if we encounter a dog – in other words, an instance of the class Dog – walking down the street, we already have a basic understanding of how to deal with it, even if we have never met that *particular* dog
Describing a Class

• We *describe* a class by listing the common features that are shared by all the objects in that class, divided into the attributes that each object *has*, and the actions that each object can *perform*.

• Example:
  • Student number is an *attribute* of the class Student.
  • Every student *has* a student number, although each individual student has a different student number.

• Example
  • Barking is an *action* that all objects of the class Dog have in common.
What is a Waiter?

- A Waiter has the following *attributes* and *actions*
  - Attributes of a Waiter
    - Name
  - Actions of a Waiter
    - Bring menus
    - Take orders
    - Bring meals
  - This collection of attributes and actions *defines* the class of Waiters
  - We can deal with any individual waiter just based on our knowledge of the class Waiter
Exercise

• Download and open the shapes BlueJ project
• Create some objects from the classes Circle, Square and Triangle (right click on the class and select new(...))
• Inspect the objects (double click on any object in the object bench)
• Invoke their methods (right click on an object and select one of its methods)
• Experiment! Making mistakes is OK - it is a good way of learning to program
What's in an Object?

- Objects have operations that can be invoked (Java calls them **methods**).
- Objects usually do something if we invoke a method.
- Objects have **state**. The state is represented by stored values in fields.
Object State

- Notice the **types** of the fields: int, String, boolean
- Types restrict the values a field can take
Instances

- Multiple instances: Many similar objects can be created from a single class.
- An object has attributes: values stored in fields.
- The class defines what fields an object has, but each object stores its own set of values (the state of the object).
A Java Method

/**
 * Move the circle horizontally by 'distance' pixels.
 */

public void moveHorizontal(int distance)
{
    erase();
    xPosition += distance;
    draw();
}
About Methods

- Methods may have parameters.
- Parameters pass additional information needed to execute the method. That is, “input” to the method.
- Parameters have types. The type defines what kinds of values a parameter can take.
- Methods may return a result via a return value.
- The head of a method is called its **signature**. It provides the information needed to invoke that method.
- Objects can **communicate** by calling each other’s methods.
Source Code

- The source code of a class determines the structure and behaviour (fields and methods) of each of the objects of that class
- A Java project (application) is a collection of classes
Return Values

- All the methods in the figures project have void return types; but …
- … methods may return a result via a return value
- Such methods have a non-void return type
- More on this in future lectures
Summary of concepts introduced

You should now be able to give an explanation of each of these terms:

object; class; method; parameter; signature; data type; multiple instances; state; method calling; source code; result

Try it! Turn to code example 1