Classes and Objects

Table of Contents

OOP overview

- All programming languages have
  - calculation
  - selection
  - iteration
  - abstraction
- **abstraction** is fundamental concept differentiating procedural from OOP languages
  - C: uses functions as unit of abstraction
    - functions manipulate data
  - OOP: combines data and function to create a class, the fundamental unit of abstraction

Classes

- Classes: generalisation of a real world entity
  - physical real world thing: student/book
  - abstract real world thing: subject
  - even more abstract thing: list/string (data
- template for things with common properties

- *attributes* and *methods*
- defines new **data type**

Objects

- **instance** of a class
- contains **state**
- **object**: specific, concrete example of a class
- **instance**: object that exists in your code
- e.g. could define Car as class, then Ford, Ferrari, Toyota may be instances of class, but dependent on the definition

Object Oriented Features

- data abstraction: creating new data types well suited to application by defining new classes
  - similar to C `struct` but with additional features i.e. attributes and methods
- encapsulation: grouping data (attributes) and methods that manipulate the data to a single entity through defining a class
  - unique to OOP, not present in procedural programming
- information hiding
- delegation
- inheritance
- polymorphism

Class definition

```
<visibility modifier> class <ClassName> {
    // attribute declarations
    <visibility modifier> <type> <variable name>;
    // method declarations
    <visibility modifier> <typeReturned> myMethod(paramList) {
        variable declarations
        statements
    }
}
```

- **instance variables**: attributes defined within class (not in methods)

  - maintain state of the object
  - **property/attribute** particular to a given object of a class

- **local variables**: variables define inside a method

Class instantiation

```
Circle aCircle;
Circle bCircle;
```

- this does not create `Circle` objects: `aCircle` is a **reference/pointer** to `Circle` objects
- currently they are **null references** as they are pointing to nothing
- **null**: Java keyword for "no object here"
- objects are `null` until **instantiated**

```
Circle circle_1 = new Circl();
Circle circle_2 = new Circl();
```

- `new`: directs JVM to allocate memory for an object, instantiating it

Garbage collection

- `circle_1 = circle_2` changes `circle_1` to point to `circle_2`
  - this leaves the original object `circle_1` referred to without any references
- an object without a valid reference (orphan) cannot be used in future
- becomes candidate for **Java automatic garbage collection**
  - periodic memory free of unused objects
  - do not need to do explicitly

Using instance variables and methods

```
<objectName>.<variableName>
<objectName.<methodName>(<args>);
```

`main` method

- a program in Java $\iff$ class with a `main` method
- `main` is void

Example: Drinking glass

- attributes
- height
- radius
- isFull
- Material: nb this could be defined as a class itself; class composition
- Shape
- methods
  - fill glass
  - empty glass
  - wash glass