O01 Classes and Objects 1

Class declaration
Object creation

Classes and objects

Java is an *object-oriented* language.

- Objects combine state (fields) and behaviour (methods).
- A class defines a type objects (what fields and methods they have).
  - Each objects is an instance of a class.
- Classes form a hierarchy.
  - `java.lang.Object` is the root (ultimate ancestor) class of all Java classes.

Class Declaration

A class declaration will have the following, in order:

- Any **modifiers** (`public`, `private`, etc.)
- The keyword `class`
- The **class' name** (first letter capitalized)
- Optional: **superclass' name** preceded by `extends`
- Optional: list of **interfaces** preceded by `implements`
- The class **body** surrounded by braces {}

Class Member Declarations

Fields and methods of a class are known as “class members”.

Field (member variable) declarations have the following, in order:

- Any **modifiers** (`public`, `private`, `static`, etc.)
- The field’s **type**
- The field’s **name**
- (optional) a ‘=’, followed by an initial value expression.

Declarations are statements – end with ‘`;`’.

Constructors

A constructor is a special method that is automatically executed when an instance is created.

Constructors differ from normal methods:

- They have **no return type**.
- They have the **same name as the class**.

If no constructor is defined, the compiler will automatically call the constructor for the class' superclass

Note: If no other constructor defined, class inherits a no-parameter constructor from `Object`.

The `this` keyword

Within instance methods and constructors, the `this` keyword refers to the object whose method or constructor is being called.

- Disambiguating field names from parameters
  - Parameters and instance field names may clash. The `this` keyword explicitly refers to the instance.
- Calling other constructors
  - When there are multiple constructors, they may call each other using `this` as if it were the method name.

Creating Objects

An object-creating expression consists of

- the keyword `new`
- followed by a call to the class' constructor

Typically, the newly created object is assigned to a variable of matching type (class).

Objects may be deleted automatically when they are known to no longer be in use (garbage collection).

Using Objects

Outside a class, an object reference followed by the dot '.' operator must be used:

- Reference the object's fields
  - Object reference, '.', field name
- Call the object's methods
  - Object reference, '.', method name, arguments in parentheses

Within instance methods, the object's fields and methods can be accessed directly by name, (optionally with the `this` keyword).

- `fieldName` or `methodName()`
- `this.fieldName` or `this.methodName()`

Overloading

A class can have several methods with the same name, but different arguments (number, type, order), often called “overloading”.

- Overloaded methods may have different return types.
- You can overload the constructor.