

A still life painting of a basket of yellow pears. The pears are rendered in various shades of yellow and green, with visible brushstrokes and highlights. They are set against a dark, textured blue background. The overall style is impressionistic, with a focus on color and light.

# O01 Classes and Objects 1

Class declaration  
Object creation

# Creating Classes and Objects

The following slides describe the *mechanics* of creating a class and creating objects (instances of that class) in Java.

Some of the mechanics *will not make much sense* until later when the relevant concepts are explained. For now, treat these as boilerplate (stuff you ‘just do’).

# 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 `{ }`

# Member Variable Declaration

Three kinds:

- Class and instance variables, called **fields**
- Variables within a method, called **local variables**
- Method arguments, called **parameters**

Member variables will have the following, in order:

- Any **modifiers** (`public`, `private`, etc.)
- The field's **type**
- The field's **name**

# 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 provided, the compiler will automatically call the constructor for the class' superclass



# Creating Objects

A statement creating an object has three parts:

- **Declaration** (a referring variable and type)
- **Instantiation** (the `new` keyword) – new object on heap
- **Initialization** (call to constructor) – initialise object

# 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()`