OO-related Java keywords

- `class`
- `interface`
- `public/private/protected`
- `extends`
- `implements`
- `abstract`
- `static`
- `final`
- `new`
- `this`
- `super`

Objects, classes, inheritance

Characteristics of objects: **state**, **behaviour**

An object is an **instance** of a general **class** of objects

In Java, a class contains **fields** (state) and **methods** (behaviour)
- **Static** fields/methods are associated with the class itself, not with an instance

Classes can **inherit** state and behaviour from other classes
- Subclass is a **specialised version** of the superclass

In Java, a class can have **exactly one** superclass
- If superclass isn't specified, then it inherits from `Object`

Subclasses can **override** superclass methods to provide specialised behaviour

Don't forget **access modifiers** (`public`/`protected`/`(default)`/`private`)

Constructors, methods

Constructor: used to create a new instance of a class (via **new** keyword)

Constructors are **not** inherited – call super-class constructor with **super** keyword

Method **overriding**: redefining method behaviour in a subclass

Method **overloading:** defining multiple methods with the same name but different signatures

Abstract classes/methods, interfaces

Abstract classes have "holes" – abstract methods that **must** be overridden

- Still have constructors, fields, normal methods, static fields/methods, etc

Final classes cannot be subclassed (e.g., for security), and final methods cannot be overridden

- **Final** fields, parameters, variables cannot have value changed after it is set
- **Static final** generally indicates class-level constants (e.g., `Long.MAX_VALUE`)

Interfaces represent class relationships **outside main inheritance hierarchy**

- Classes **implement** interfaces – can implement any number of them (including zero)
- All methods implicitly **public abstract**; all fields **public static final**
- Support multiple inheritance of **type** (not of state or of implementation)