Australian National University

# Classes and Objects 1

01

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 Components, in Order

1. Any **modifiers** (`public`, `private`, `static`, etc.)
2. The keyword `class`
3. The class' **name** (first letter capitalized)
4. Optional Type Parameters surrounded by angle brackets `<>` (J12)
5. Optional **Superclass Declaration** (`extends` *[superclass]*)
6. Optional **Super-Interface list** (`implements` *[list-of-interfaces]*)
7. The **class body** surrounded by braces `{}`

# Class Body

The body of a class lists its members, which are any of the following:

- Class and instance variables, called (static) fields
  - Modifiers (`public`, `private`, `static`, etc.), type, and name, in that order
- Class (static) and instance methods
  - See J7
- Constructors
  - Up next
- Nested classes and interfaces, enums, etc. (J7)

In addition, a class can have class (static) and instance initializer blocks (O2)

# Constructors

Special methods to initialize objects/instances when they are created.

Differences to normal methods

- **No return type**
- **Same name as the class**

When no other constructor is provided, default constructor has no parameters and requires superclass to have a constructor without parameters, too.

# Declaring and Creating Objects

Variable *Declaration*

- Modifiers + type + variable name

*Instantiation*

New-expression

1. Keyword **new**
2. Class **name**
3. Optional **Generic Type Arguments** surrounded by angle brackets <> (J12)
4. **Arguments** surrounded by parentheses

*Initialization*
(call to constructor)

```
MyClass myObject = new MyClass(myArg1);
```

# Using Objects

Inside a class, you can generally refer to members by their name.

Outside, or when more closely nested definitions “shadow” a member’s name, use the dot ‘.’ operator:

[receiver] . [name]

[receiver] . [name] < [type arguments] > ( [arguments] )

**Receivers**

- For instance members, a reference to the object (inside instance methods/constructors, the special reference `this` is available)
- For class members, the name of the class (plus type arguments, if any)