Classes and Interfaces

Kevin Jones

@kevinrjones

Overview

- Declaring classes
- Creating instances
- Properties
- Members
- Declaring interfaces
- Interface details
- Implementing interfaces
- Inline classes

Classes

Defined with the ‘class’ keyword

public by default

final by default

Have a primary constructor

May have secondary constructors

No ‘new’ keyword

Maybe abstract

May have properties and methods

Classes

Classes represent things

- Planets
- People

Classes derivation model an ‘isA’ relationship

- A *Planet* ‘isA’ *SpaceBody*
- A *HabitablePlanet* ’isA’ *Planet*

Interfaces

Very similar to abstract classes

Generally are purely abstract

- Although can have default implementations

Cannot store state

Interfaces

Interfaces represent ‘actions’

- Database connections
- Logging

Interfaces model a ‘hasA’ relationship

- A *Person* ‘hasA’ *Logger* calls
- A *Planet* ‘hasA’ *DatabaseConnection*

Summary

Declaring classes

- Use the ‘class’ keyword
- Has a primary constructor
- May have secondary constructors
- May have an init method

Creating instances

- No ’new’ keyword
- Just call the constructor like it’s a method

Summary

Properties

- Class may have properties
- Can be read-only
- Must be initialized
- May be late-initialized

Members

- Class may have members

Overriding

- Class must be ‘open’ or ‘abstract’
- Methods/properties must be ‘open’
- Use the ‘override’ keyword in deriving class

Summary

Declaring interfaces

- Declared with the ‘interface’ keyword
- Essentially abstract
- Used to model a ‘hasA’ relationship

Implementing interfaces

- Implemented by a class
- Implements all methods

Summary

Inline classes

- Declared with the ‘value’ keyword
- And the ‘JvmInline’ annotation
- Have a single property
- Compile down to the type of the property
- Efficient and carry more meaning

Data Classes