Inheritance in Kotlin

CO5225 – Andrew Muncey

Contents

- Making classes inheritable
- Primary constructors
- Overriding methods
- Interfaces
- Abstract classes

Inheritance

- By default classes in Kotlin **cannot** be inherited
  - Equivalent of using the final keyword for a Java class
- To make a class inheritable requires the **open** keyword
  - This applies with chains of inheritance, i.e. any class which is inherited must also be declared as open

```
open class BaseClass {

}

class Subclass : BaseClass() {

}
```

Inheritance with primary constructor

- If the superclass has a primary constructor it must be called when the subclass is created

```
open class BaseClass(val x: Int){

}

class Subclass(x: Int) : BaseClass(x){

}
```

Overriding methods

- In the same way as a class declaration, methods must be marked as open in order to be overridden
- Methods in the base class are called using the the keyword ‘super’

Interfaces

- Essentially a set of unimplemented functions that must be implemented by anything that adheres to that interface
- May provide a default implementation of a method
- May contain properties, but these cannot have backing fields
- An Interface can derive from (multiple) other interfaces
- Does not have constructors
- A class can implement multiple interfaces

Abstract class

- A class which contains abstract methods
- Abstract methods are unimplemented (like those in an interface)
- Abstract class may contain methods which are implemented
- Open by definition
- Only one class (abstract or otherwise) can be inherited by any given class
  - A chain of inheritance is permitted
- Can inherit from existing open classes