Java Interfaces and Abstraction

INTRODUCING JAVA ABSTRACTION MECHANISMS

Richard Warburton

@richardwarburto www.monotonic.co.uk

Problems with Revenue Calculator

Code all in one Place

Doesn’t scale, hard to locate

Hard to extend

Need to modify the code calculator to change anything

Hard to configure

Stringly Typed method variable with no way to alter rates

Course Modules

Understanding the Need for Abstractions

Extending Your Code Through Interfaces

Implementing Interfaces in Different Modules

Recognizing the Dangers of Over-abstraction

Abstract Class

A class that can have abstract methods that cannot be directly instantiated.

```
abstract class RevenueCalculator

{}


class HourlyRateCalculator extends RevenueCalculator

{}
```

Abstract Classes

Interface

A named set of methods. A class implements an interface, meaning that it has those methods.

```
interface RevenueCalculator
{}

class HourlyRateCalculator implements RevenueCalculator
{}
```

Interfaces

Methods

Abstract Classes	Interfaces
Methods without keywords have bodies	Methods without keywords don’t have bodies
`abstract` keyword lets you remove the body	`default` keyword lets you add a body
Methods can be public, private, protected or package-private – the default	All methods are public

Fields

Abstract Classes

Can have fields.

Non-private fields visible in subclasses.

Interfaces

Cannot have instance fields.

No sharing of state.

Inheritance

Abstract Classes

Single Inheritance

Interfaces

Multiple Inheritance

Polymorphism

Objects of a child class can be referred to by their parent's class, methods called on the parent bind to the child's implementation.

Demo

Refactor earlier example

- With abstract classes
- With interfaces

Either way we introduce an abstraction

In the next module we'll use it

Prefer Interfaces to Abstract Classes most of the time

Summary

Interfaces and abstract classes are Java language abstraction tools

Enables polymorphism and abstraction