

Inheritance

Abstract Classes and Interfaces



Motivation

- Problem
 - Want a common design for subclasses
 - Able to provide some implementations, but not all
- Solutions
 - Have method return `null`, hope overridden later
 - Create an `abstract` method, force overriding later



Abstract Classes

- Any class that contains abstract methods
 - Subclasses MUST override all abstract methods
- May also contain non-abstract methods and members
- May not be instantiated, but can be referenced
 - Unable to create an actual object of that class
 - Able to reference using upcasting or downcasting

<http://docs.oracle.com/javase/tutorial/java/landl/abstract.html>



Abstract Classes

- A constructor may not be abstract
 - Constructors may NOT be overridden
 - Abstract methods MUST be overridden
- A static method may not also be abstract
 - If static, can access via class name
 - If abstract, no implementation through that class



Polygon Example

- All polygons have a list of points
 - Subclasses will initialize different number of points
- All polygons can be drawn using same method
 - Provide non-abstract `draw()` method
- All polygons have different area functions
 - Provide abstract `area()` method



Motivation

- Problems
 - Want consistent design for subclasses
 - Unable to provide any implementations
 - Can only inherit directly from one superclass
- Solution
 - Use an interface instead of a (abstract) class



Interfaces

- Provide a consistent *interface* for interaction
 - Uses `interface` instead of `class` keyword
 - Think as a *lightweight* class
- Has only constants members (implicitly `public`, `static`, and `final`)
- Has only abstract, `static`, or default methods (implicitly `public`)

<http://docs.oracle.com/javase/tutorial/java/andl/createinterface.html>



Interfaces

- Java 8 introduced **default methods** for interfaces
 - Non-abstract method with a default implementation
- Also allows **static methods** in interfaces
 - Essentially a default method that does not access any other interface methods

<https://docs.oracle.com/javase/tutorial/java/land/defaultmethods.html>



Interfaces

- Can implement as many interfaces as needed
- Can extend a class and implement one or more interfaces simultaneously
- Can extend an interface to create interface hierarchies
 - See `Collection` hierarchy

<http://docs.oracle.com/javase/tutorial/java/landl/createinterface.html>



Abstract Classes vs Interfaces

- Abstract Classes
 - Implementations and instance members allowed
 - Unable to extend multiple classes
- Interfaces
 - No instance members, limited method options
 - Able to implement multiple interfaces





CHANGE THE WORLD FROM HERE