10

Abstract Classes and Interfaces: Part II

Last time

- review abstract classes and interfaces
- multiple interfaces
- event-driven programming
- GUI program: mouse and keyboard event handling

Objectives

- review event-driven programming
- `instanceof` keyword
- `final` keyword with classes and methods
- GUI programming practice

OOP Concepts

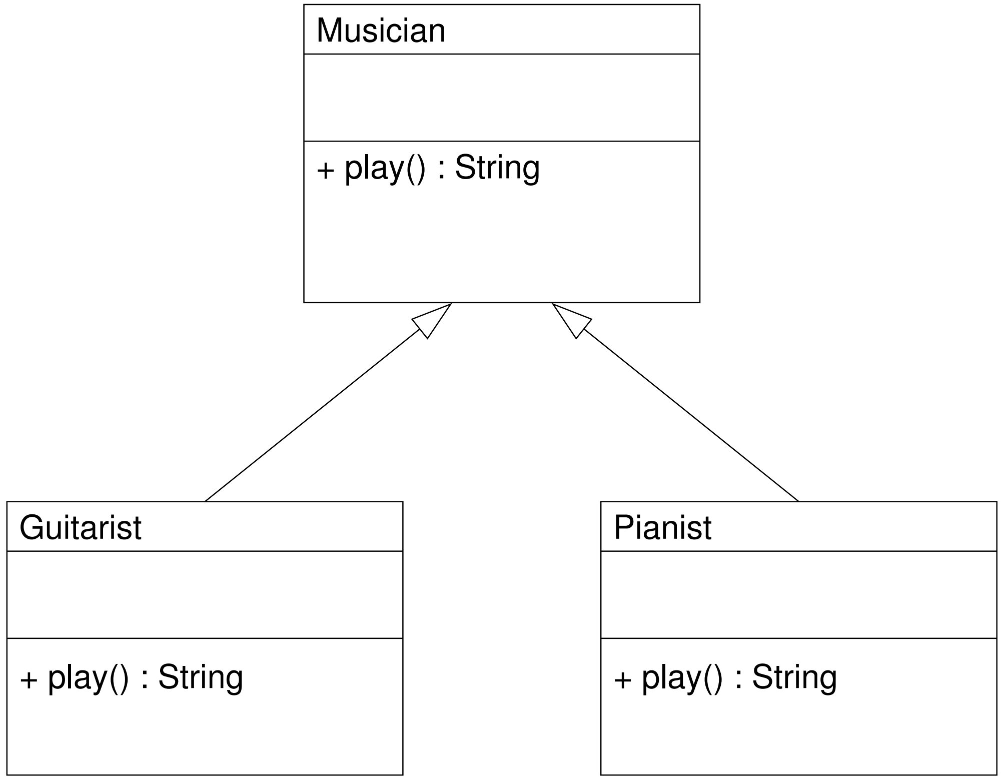
- Inheritance (`extends`)
- Abstract classes and methods (`abstract`)
- Interfaces (`interface, implements`)
- Polymorphism (*poly-morph*: many forms)

OOP Syntax example

```
public class ClassName
     extends SuperclassName
     implements FirstInterface,
                SecondInterface, ...
{
     ...
}
```

Example: Inheritance hierarchy

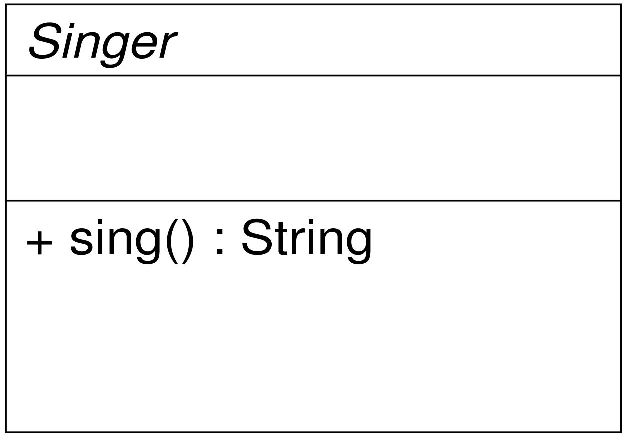
Implement the `Musician` class hierarchy shown below.



Example: Singer interface

Define the `Singer` interface shown below.

Implement the `Singer` interface sing method for the `Guitarist` class to return the string “Singing a rock song.”



Multiple interface implementations

- Class can implement as many interfaces as needs
  - Use a comma-separated list of interface names after keyword implements
  - Example:

```
public class ClassName extends SuperclassName
implements FirstInterface, SecondInterface, …
```

Operator `instanceof`

- Dynamic binding
  - Also known as late binding
  - Calls to overridden methods are resolved at execution time, based on the type of object referenced
- `instanceof` operator
  - Determines whether object is an instance of a certain type

Class `Class`

- `getClass` method
  - Inherited from `Object`
  - Returns an object of type `Class`
- `getName` method of class `Class`
  - Returns the full name of object’s class

Downcasting

- **Convert superclass reference to a subclass**
- **Allowed only if the object has an *is-a* relationship with the subclass**

`final` Methods and Classes

- `final` methods
  - Cannot be overridden in a subclass
  - `private` and `static` methods are implicitly `final`
  - resolved at compile time, this is known as static binding
- `final` classes
  - Cannot be extended by a subclass
  - All methods in a `final` class are implicitly `final`