Interfaces and Polymorphism

Chris La Pilla
CompSci 6
Summer 2007

Making code more general

- We frequently have situations in which several different classes have a similar set of methods
- Example: Circle class and Rectangle class we implemented both have methods for computing area and perimeter
- We’d like to write code that could work with any of the classes

Interfaces

- An interface type is a list of methods (without implementation)
- A class that conforms to an interface is said to implement the interface (i.e. it provides an implementation for all of the interface's methods)
- Multiple classes may implement the same interface
- A class can implement multiple interfaces

Creating an interface

- An interface is like a class declaration, except:
  - All methods are abstract (name, return type, and parameter list), but no implementation
  - All methods are automatically public
  - There are no instance variables

Interface

```java
public interface MeasurableShape{
    double getArea();
    double getPerimeter();
}
```

```
public class Circle implements MeasurableShape{
    ...
}
```

Interface types and variables

- Variables can have an interface type:
  ```java
  MeasurableShape shape;
  ```
- The variable can hold a reference to an object of any class that implements the interface
  ```java
  shape = new Rectangle(...);
  shape = new Circle(...);
  ```
- Idea: rather than writing code for each specific class, we write code that works with any object that implements the interface
Polymorphism

• If a variable's type is an interface, we can only call methods that are part of that interface
• The actual method that is executed is the one that belongs to the class of the object that is referenced by the variable
• In other words, the right method is always called, even though the calling code doesn't know which type of object the variable actually references!
• This is known as polymorphism

Example

• Suppose `Rectangle` and `Circle` both implement `MeasurableShape`

  ```java
  MeasurableShape shape = new Rectangle(...);
  double area = shape.getArea();
  // computes area as width * height

  shape = new Circle(...);
  area = shape.getArea();
  // computes area as PI * r * r
  ```