## CSE 142 Summer 2001

Interfaces



## Introduction

- Review
  - Classes
  - Types
- Today
  - Interfaces – type specifications



## Review – Class Declarations

- A class declaration specifies two things
  - A *type definition*: the class name is a new type; the class declaration specifies the member functions and data associated with the type
  - An *implementation*: code for methods belonging to the class (also initialization code sometimes)
- It is sometimes very useful to be able to specify a type without including implementation details



## Interfaces

- An interface defines a new type but gives no implementation
- Interface may contain method specifications and symbolic constants *only*. Example:

```
/** Interface to collection of weather data */
public interface Weather {
    /** add daily weather record to this collection */
    public void add(DailyWeather d);
    /** print daily weather records */
    public void printRecords( );
    /** return total rainfall */
    public double totalRain( );
}
```



## Implementing Interfaces

- To create instances of an interface, the interface must be implemented by some class

```
/** Implementation of Weather interface */
public WeatherImpl implements Weather {
    /** Construct empty WeatherImpl object */
    public WeatherImpl( ) { ... }
    /** add daily weather record to this collection */
    public void add(DailyWeather d) { ... }
    /** print daily weather records */
    public void printRecords( ) { ... }
    /** return total rainfall */
    public double totalRain( ) { ... }
}
```



## Using Interfaces

- A client can declare names with the interface type
- These names can refer to any object whose class implements the interface

```
Weather monthlyWeather = new WeatherImpl( );
monthlyWeather.add(new DailyWeather("nice", 72, 68, 0.0));
```

- Reminder: can't create an instance of the interface directly

```
Weather naughtyWeather = new Weather( );          // doesn't work (why?)
```

## Some Details

- If a class implements an interface, it must provide implementations of *all* methods defined in the interface (can't implement just a subset)
- Classes may implement more than one interface
- Classes may (often do) contain additional methods and fields besides the ones specified in the interface(s) they implement



## Key Use for Interfaces

- Different classes may implement the *same* interface
  - Different implementations may have different characteristics (space efficiency, execution time)
  - Client code can be written in terms of the interface, then can use different implementations as desired
- Widely used in the Java Standard libraries
- Useful for organizing our own code
- Example: from class GWindow
  - Classes Rectangle, Oval, Triangle and Line all implement Shape

```
/** add a shape to this GWindow */
public boolean add(Shape s) { ... }
```