Abstract Methods, Abstract Classes, and Interfaces

Review Weiss, sections 4.2 through 4.5 and section 4.7 (pages 125–141 and 150–157)

I am indebted to my colleague, John David Stone, for assistance with this handout.

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/
or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.
Abstract Method

• Different derived classes implement the abstract method differently.

```java
public static double totalArea( WhatType [] arr )
{
    double total = 0.0;
    for( int i = 0; i < arr.length; i++ )
        if( arr[ i ] != null )
            total += arr[ i ].area();

    return total;
}
```
public abstract class Shape {
    int x, y;
    public abstract double area();
    public Shape( int x, int y ) { this.x = x; this.y = y; }
}

public class Circle extends Shape {
    private double radius;
    public Circle( double rad ) {
        super( 4, 5);
        radius = rad;
    }
    public double area() {
        return Math.PI * radius * radius;
    }
}
Abstract Method (cont.)

```java
public static double totalArea( Shape [] arr )
{
    double total = 0;

    for( Shape s : arr )
        if( s != null )
            total += s.area( );

    return total;
}
```

```java
public static void main( String [] args )
{
    Shape [ ] a = { new Circle( 2.0 ), new Rectangle( 1.0, 3.0 ), null };
    System.out.println( "Total area = " + totalArea( a ) );
    printAll( a );
}
```
Multiple Inheritance

• Cause conflict implementations and fields

• Interface
  – Consists of public abstract methods and public static final fields
  – A subclass implements all the abstract methods.
  – A subclass can implement multiple interfaces.

```java
Student x) { x.work(); }
Employee x) { x.work(); }
```

```java
t( new StudentEmployee( ) );
u( new StudentEmployee( ) );
```