Interfaces

Reading for today: Sec. 12.1 and corresponding ProgramLive material. Also, compare with previous discussions of abstract classes (Sec. 4.7).

Final exam: **Period B: Thursday, 13 May, 9:00-11:30AM, Barton East**
Contact mwitlox@cs.cornell.edu ASAP regarding conflicts. Explain the nature of the conflict in detail (e.g. which other exam it conflicts with).

Can extend only one class

public class C extends C1, ~~C2~~ { … }

```
public class C1 {
   public int m() {
      return 2;
   }

   public int p() {
      return …;
   }
}
```

```
public class C2 {
   public int m() {
      return 3;
   }

   public int q() {
      return …;
   }
}
```

if we allow multiple inheritance, which m is used?

Can extend only one class

public class C extends C1, ~~C2~~ { … }

```
public abstract class C1 {
   public abstract int m();
   public abstract int p();
}
```

```
public abstract class C2 {
   public abstract int m();
   public abstract int q();
}
```

But this would be OK, because the
bodies of the methods are not given!
Nevertheless, not allowed

Use an "interface"

public class C **implements** C1, C2 { … }

```
public interface C1 {
   int m();
   int p();
}
```

```
public interface C2 {
   int m();
   int q();
}
```

Methods declared in an interface must be abstract!
No need for "abstract", automatically public

A use of interfaces: remove need for duplicate code for special cases of a general approach

Example: sorting is general, but the notion of "<" may change:

Recommender systems should sort products (movies, songs …) by quality or by how much you, personally, would like them.

Travel sites should sort flights by price, departure, etc.

Don't want to write many sort procedures:

```
public void sort(int[] arr) {…}
public void sort(double[] arr) {…}
public void sort(Movie[] arr) {…}
public void sort(Flight[] arr) {…}
```

Use an interface to enforce the existence of a method that does the comparison of two objects

Interface java.util.Comparable

```
/** Comparable requires method compareTo*/
public interface Comparable {
   /** = a negative integer if this object < c,
       = 0 if this object = c,
       = a positive integer if this object > c.
       Throw a ClassCastException if c cannot
       be cast to the class of this object. */
   int compareTo(Object c);
}
```

abstract method: body replaced by ;

Every class that *implements* Comparable must override compareTo(Object).

Classes that implement Comparable
Boolean
Byte
Double
Integer
…
String
BigDecimal
BigInteger
Calendar
Time
Timestamp
…

Using an interface as a type

```
/** Swap b[i] and b[j] to put larger in b[j]  */
public static void swap( Comparable [] b, int i, int j) {

        if (b[j].compareTo(b[i]) < 0) {
                Comparable temp= b[i];
                b[i]=  b[j];
                b[j]=  temp;
        }
}
```

Polymorphism: the quality or state of existing in or assuming different forms

This parametric polymorphism allows us to use swap to do its job on any array whose elements implement Comparable.

7

```
/** An instance is a movie and what critics thought of it. */
public class Movie implements Comparable {
   public String name;  /** movie name. */
   private int[10] ratings; /**ratings from 10 critics. */
   private int final NOTSEEN= 0; /** rating if not seen by given critic */
   /** Actual ratings are: 1, 2, 3, 4, or 5 */

   /** = -1, 0, or +1 if this Movie's name comes alphabetically before,
         at, or after c.
      Throw a ClassCastException if c cannot be cast to Movie.*/
   public int compareTo(Object c) {
      if (!(c  instanceof  Movie))
         throw new ClassCastException("argument is not a Movie");

      // Note: String implements Comparable
      Movie cm= (Movie) c;
      return this.name.compareTo(cm.name);

}
```

class will contain other methods

8

Another example: Listening to a mouse click (or other object-appropriate action)

Defined in package java.awt.event

```
public interface ActionListener extends Eventlistener {
   /** Called when action occurs. */
   public void actionPerformed(ActionEvent e);
}
```

```
/** An instance has two buttons. Exactly one is always enabled. */
public class ButtonDemo1 extends JFrame
                          implements ActionListener {
  /** Process a click of a button */
  public void actionPerformed (ActionEvent e) {
          boolean b= eastB.isEnabled();
          eastB.setEnabled(!b);
          westB.setEnabled(b);
  }
}
```

9

Declaring your own interfaces

```
/** comment*/
public interface <interface-name> {
   /** method spec for function*/
   int compareTo(…);

   /** method spec for procedure */
   void doSomething(…);

   /** explanation of constant x*/
   int x= 7;

}
```

Use ";" instead of a body

Methods are implicitly **public**. You can put the modifier on if you wish.

Every field is implicitly **public**, **static**, and **final**. You can put these modifiers on them if you wish.

10

A class can implement several interfaces

```
/** comment*/
public class C implements Inter1, Inter2, Inter3{
   …
}
```

The class must override all methods declared in interfaces Inter1, Inter2, and Inter3.

Example: a recommendation system that returns all movies that satisfy some minimum similarity to one of your favorites.

Need to sort *and* to measure similarity (a general task worthy of an interface).

11