Interfaces – handling very “general” operations cleanly (April 21)

Reading for today: Sec. 12.1 and corresponding ProgramLive material. It might also be useful to compare with our previous discussions of abstract classes (Sec. 4.7).
Reading for next time: section 9.3 (arrays of arrays)

Graded prelim IIIs available up front. After today, they can be retrieved from Upson 360, M-F 10am-noon and 2-4pm; bring ID.

Assignment A7 is due Thursday the 30th.

The labs next week are optional, and will simply serve as office hours. (This week’s lab is **not** optional.)

The final exam is Friday May 8th, 9:00am-11:30am, Barton Hall (west side). Please contact mwitlox@cs.cornell.edu ASAP regarding conflicts.

Motivation: don’t duplicate code for special cases of a general approach

Example: sorting is very useful for information analysis and display of all kinds of things:

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.

But we don’t want to write both sort(Movie] arr) and sort(Flight[] arr).

Java.util.Arrays.sort(Object[] arr) to the rescue!...

…**if** the objects in arr **implement** the Comparable **interface**, which enforces the existence of a *general* function compareTo() that can serve the role of the more specific operators <, ==, and > we’ve been using in our sorts.

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);
}
```

An 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
…

```
/** 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 a certain 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");

    // String implements Comparable
    return this.name.compareTo(((Movie) c).name);

}
```

class will contain other methods

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);
  }
}
```

5

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.

6

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).

7

You can use 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.

8