

Lecture 20: Interfaces

- Last time:
1. Explicit array initialization
 2. Arrays as arguments
 3. Arrays of objects
 4. Privacy leaks
 5. Array Copying

- Today:
1. Interfaces
 2. Wrappers



CMSC 131 Fall 2007
Jan Plane (Adapted from Bonnie Dorr)

Code Re-use

- Many operations recur in programming
 - sorting
 - max / min
(These operations may apply to strings, numbers, etc.)
- Desirable: one implementation!
 - Less coding
 - Less likely to have typos
 - Easier maintenance of code

CMSC 131 Fall 2007
Jan Plane (Adapted from Bonnie Dorr)



Polymorphism

- Using an **interface** we can create one variable that can reference objects of different types (i.e. UMStudent variable referencing CSMajor, CEMajor or PsychMajor)
- This form of "generalization" is called **polymorphism**
 - Hallmark of OO languages
 - Allows application of same code to objects of different types
 - Polymorphism: "A variable that takes on many shapes."
- Interfaces: one mechanism Java provides for polymorphism
 - a collection of prototypes (method prototypes but no bodies) aka **abstract methods**
 - A class C **implements** an interface I
 - if C provides implementations of all of I's abstract methods
 - A class implementing an interface can also provide other methods or implement other interfaces

CMSC 131 Fall 2007
Jan Plane (Adapted from Bonnie Dorr)



Wrappers



- We may want to treat primitives as though they were objects
- For example, generic routines (like `PsychoAnalyze`) can be implemented using interfaces ... but they are not usable on primitive types
- To overcome this problem, Java provides **wrappers** for primitive types
 - Wrappers: classes whose objects contain single values of the "wrapped type"
 - Wrappers also contain other useful conversion operations (to / from `String`, etc.)
 - Wrappers included in `java.lang`:
 - `Byte`
 - `Short`
 - `Integer`
 - `Long`
 - `Float`
 - `Double`
 - `Character`
 - `Boolean`

The Integer Wrapper



- The documentation is on-line at <http://java.sun.com/j2se/1.5.0/docs/api/>
- Notes
 - Constructors
 - Implements `Comparable`
 - Documentation says "`Comparable<Integer>`"
 - `Comparable` in Java 5.0 is a interface
 - Has `compareTo` method, etc.

In class Demo: Implementing a method using the Integer class



- Create objects of type `Integer`
 - using the constructor
 - can be based on `int` type values or variables
- Create an array of `Integer` type object references and those objects of type `Integer`
- use the API to access information about the data in the `Integer` class
