

Lecture Set #12: Polymorphism Introduction

1. Wrappers
2. Interfaces



CMSC 131 Spring 2009
Jan Plane (Adapted from Bonnie Dorr)

Wrappers

- We may want to treat primitives as though they were objects
- For example, generic routines 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

CMSC 131 Spring 2009
Jan Plane (Adapted from Bonnie Dorr)

1

The Integer Wrapper

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

CMSC 131 Spring 2009
Jan Plane (Adapted from Bonnie Dorr)

2

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 Spring 2009
Jan Plane (Adapted from Bonnie Dorr)

3

Polymorphism



- Using an **interface** we can create one variable that can reference objects of different types (i.e. UMSStudent 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 Spring 2009
Jan Plane (Adapted from Bonnie Dorr)

4

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
- Expand this example to Strings
- Expand this example to Cats

CMSC 131 Spring 2009
Jan Plane (Adapted from Bonnie Dorr)

5

Adapting Cat to Implement Comparable



- The Comparable Interface
 - insists that I must implement `compareTo` method which has the following prototype:

```
int compareTo(Object o)
```
 - it must return a negative if the current object is less, a positive if the current object is greater or a 0 if they are the same.
- What is `Object`?
 - Type of all possible objects in any class
 - Shortcoming of (earlier) Java: no good way to say "same type as `this`"
 - Instead: Implementation must take any object

CMSC 131 Spring 2009
Jan Plane (Adapted from Bonnie Dorr)

6

What about `int`? `char`?



- Polymorphic `findMin` can be used on any class implementing `Comparable`
- What about primitive types (`int`, `char`, `double`, etc.)?
 - They are not classes
 - They do not implement `Comparable`
 - Hence `findMin` cannot be used on them
 - That's why we use wrappers!

CMSC 131 Spring 2009
Jan Plane (Adapted from Bonnie Dorr)

7
