CMSC 132: OBJECT-ORIENTED PROGRAMMING II

Java Language Constructs I

Department of Computer Science
University of Maryland, College Park

Announcements

- Regarding questions over e-mail
  - Due to the large number of students in class, we (instructors and TAs) cannot address project questions, questions about lecture material, etc. over e-mail. If you have any questions, please address them in lab/discussion session, in lecture or during office hours. Thank you for your cooperation. ☺
- About Terpconnect
  - Your first project requires a terpconnect account and this may take at least 24 hours. No project extensions will be granted due to terpconnect account problems. It is your responsibility to start the project ahead of time.
- About Quizzes
- Please read the guidelines at:
- http://www.cs.umd.edu/~nelson/classes/utilities/examRules.html
- Regarding recommendations

About Style

- Let’s go over the “Style Guidelines” in the resources section of the class web page

Enumerated Types

- New type of variable with set of fixed values
  - Establishes all possible values by listing them
  - Supports values(), valueOf(), name(), compareTo()…
  - Can add fields and methods to enums
- **Example:** Color.java
- In Eclipse we define them as we defined classes
- When to use enums
  - Natural enumerated types – days of week, phases of the moon, seasons
  - Sets where you know all possible values
- **Example:** Deck.java
- **Example:** BoardCell.java

Implementing Equals

- Approach we want to use (assuming class A)

```
public boolean equals(Object obj) {
        if (obj == this)
            return true;
        if (!(obj instanceof A))  //  covers obj == null case
            return false;
        A a = (A)obj;
    /* Specific comparison based on A fields appears here */
}
```

- What happens if we use comparisons of Class objects rather than instanceof?
- **Example:** equalsMethod package

Comparable Interface

- Comparable
  - public int compareTo(T o)
  - a.compareTo(b) returns
    - Negative if a < b, 0 if a == b, positive if a > b
- Properties
  - Referred to as the class's *natural ordering*
  - Can sort using Collections.sort( ) & Arrays.sort( )
    - Example: Collections.sort(myList);
  - Can use as keys in SortedMap & SortedSet
  - Consistency w/ equals( ) strongly recommended
    - x.equals(y) if and only if x.compareTo(y) == 0
- **Example:** comparableExample package

Annotations

- Annotation – Provides data about a program with not direct effect on the operation of the code they annotate
- Uses
  - Information for the compiler (e.g., suppress warnings)
  - Compiler/Deployment time processing
    - Tools can process annotations in order to generate code
  - Runtime
    - Some are available to be examined at runtime.
- Validity constraint examples
  - A instance variable cannot assume a negative value
  - A parameter can not be null
  - A method in a class must override a method in its superclass

Annotations

- In JUnit4 we use @Test to identify an annotation
- Syntax

  at-sign (@) followed by annotation type and a parenthesized list of element-value pairs (no parentheses needed if not elements are present)
- Annotations used by the compiler
  - **@Deprecated** – Element is deprecated and should no longer be used
  - **@Override** – Informs compiler element is meant to override an element. If the method does not correctly override a method, a compiler error will be generated
  - **@SuppressWarnings** – Informs the compiler to suppress specific warnings
- Reference
  - http://docs.oracle.com/javase/tutorial/java/annotations/basics.html