# CMSC 132: Object-Oriented Programming II


## Java Language Constructs I

Department of Computer Science
University of Maryland, College Park

# About Style

Questions about style: two good references:

- http://www.cs.umd.edu/class/fall2013/cmsc132h/resources.shtml
- http://www.cs.umd.edu/class/fall2013/cmsc132h/styleGuidelines.html

# 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

# 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

# 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

# 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/javaOO/annotations.html