Announcements

- CMSC132 Notes - Written by former student/TA
- List of shortcuts in Eclipse
  - (CTRL + SHIFT + L)
- Adding System.out.println()
  - Type `sysout` followed by CTRL + SPACE
- CTRL + SHIFT + F
  - To format your code
- Adding 80-Characters Mark
- You are responsible for checking announcements we post in Piazza
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, ColorDriver.java
- In Eclipse we define them as we define classes
- When to use enums
  - Natural enumerated types - days of week, phases of the moon, seasons
  - Sets where you know all possible values
- **Example:** Rank.java, Suit.java, Card.java, CardDriver.java
- **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))  // handles 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: equalsComparable 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:** equalsComparable package
About Style

• Let’s go over the “Java Style Guide” in the Resources section of the class web page
Annotations

• Annotation - Provides data about a program with no 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
  • An 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 no 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](http://docs.oracle.com/javase/tutorial/java/annotations/basics.html)
Comparing Files In Eclipse

- Select the files
- Right-click and select “Compare With” → “Each Other”
- You can check your output against files with expected output by
  - Adding a System.out.println to the public test
  - Saving the results in a text file
  - Comparing using the above method
- Online sites to compare text files can be found in the Resources → Other section of the class web page
Eclipse Errors/Warnings Settings

• The following settings could help you:

http://www.cs.umd.edu/eclipse/other/#editing