CS211: COMPARABLE

1. Comparison

1.1 Classifications
• need to validate something
• make choices based on validation
• fundamental for selection
• classifications of comparisons:
  - identity
  - quantity

1.2 Identity
• comparison of primitives: ==, !=
• comparison of objects: equals
• equals method defined in class Object
• you can define your own equals methods!
• discussed more in inheritance

1.3 Quantity
• primitives:
  - compare values... very common!
  - use relation ops: <, >, <=, >=
• objects?
  - example) compare wrapper objects
  - example) compare Strings
  - example) compare People
  - cannot use relops!
  - use Comparable interface!
• example)
  class Person implements Comparable {
    // code not shown
  }

In some other method:
Person p1 = new Person(10);
Person p2 = new Person(20);
System.out.println(p1.compareTo(p2));

2. Comparable Interface

2.1 Interface Reminder
• implement interface to make a class a subtype
  class Blah implements Blurtalble
• inside Blah must implement all methods that Blurtalble defines:
  interface Blurtalble {
    void buh(int x);
  }
class Blah implements Blurtalble {
  void buh(int x) {
    System.out.println(x);
  }
}
• class Blah may have many more methods

2.2 Making Objects Comparable
• implement the Comparable interface
  class Person implements Comparable {
    // code
    // must implement compareTo method
  }
• by definition, has only compareTo method:
  int compareTo(Object o)
• current.compareTo(supplied):
  - if current < supplied, return negative int
  - if current == supplied, return 0
  - if current > supplied, return positive int
• strongly recommended that equals method and compareTo be consistent (from API):
  (x.compareTo(y)==0) == (x.equals(y))
• more specs in API
  http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Comparable.html
3. Person Example

3.1 Compare by weight

class Person implements Comparable {
    private int weight;
    public Person(int w) { weight = w; }
    public int compareTo(Object o) {
        Person p = (Person) o;
        return weight - p.weight;
    }
    public boolean equals(Person p) {
        return p.weight==weight;
    }
    // could say (Person o).weight
    // why the (Person o)?
}

3.2 Test

public class PersonExample {
    public static void main(String[] args) {
        Person p0 = new Person(100);
        Person p1 = new Person(100);
        Person p2 = new Person(200);
        Person p3 = new Person(300);
        System.out.println(p0==p1);
        System.out.println(p0.equals(p1));
        System.out.println(p1.compareTo(p2));
        System.out.println(p3.compareTo(p1));
    }
}

4. Complex Number Example

4.1 How to compare?

• rem: sqrt(-1) = i (or j)
• examples:
  - 1 < 2 (yes)
  - 2 > 1 (yes)
  - 1 == 1 (yes)
  - 1+3i < 1 + 4i (probably)
  - 1 + 2i ? 2 + 1i (?)
• should equals return the same as compareTo?
  • http://www.purplemath.com/modules/complex.htm

4.2 Comparable complex numbers

class Complex implements Comparable {
    private double real;
    private double complex;
    public Complex(double real, double complex) {
        this.real = real;
        this.complex = complex;
    }
    public String toString() {
        String op="+";
        if(complex<0) op="-";
        return real+op+complex+"i";
    }
    public boolean equals(Complex c) {
        return complex==c.complex && real==c.real &&
            mag(this)==mag(c);
    }
    public int compareTo(Object o) {
        Complex c = (Complex)o;
        if (mag(this) < mag(c)) return -1;
        else if (mag(this) > mag(c)) return 1;
        return 0;
    }
    private double mag(Complex c) {
        double asqu = c.real*c.real;
        double bsqu = c.complex*c.complex;
        return Math.sqrt(asqu+bsqu);
    }
}

4.3 Testing

public class ComplexTest {
    public static void main(String[] args) {
        Comparable c1 = randomComplex(1,3);
        Comparable c2 = randomComplex(1,3);
        System.out.println("C1: " + c1);
        System.out.println("C2: " + c2);
        System.out.println(c1.compareTo(c2));
    }
    private static Complex randomComplex(int x, int y) {
        return new Complex(myRandom(x,y),myRandom(x,y));
    }
    private static int myRandom(int min, int max) {
        return (int) (Math.random()*(max-min+1)) + min;
    }
}