# Enumerated Types – 01/24/2012

Choose one person to MANAGE the exercises and keep the team on track. Turn in all of the team's worksheets into the folder at the end of class.

### Exploring the Model

In the Gaddis book, look at the example on page 580, EnumDemo.java:

a. What is the name of the enum data type?

b. What are the allowable values?

c. What are these values (in other words are they variables, constants, objects, primitive types…what do you think they are based on what you see)?

d. Which value is the lowest possible value? Highest?

e. What method lets us compare enumerated type values?

f. Write a statement that declares choreDay as a Day type.

g. Write a statement that assigns choreDay the value for Saturday.

h. Write an expression that returns `true` if choreDay is greater than workday and `false` otherwise.

i. Think of a way that you might use the ordinal method of the `enum` type.

j. In your own words, what are enumerated data types?

### Extending the model

k. Look at the following code for Creature.java.

```
// Declare the Creature enumerated type.
enum Creature { HOBBIT, ELF, DRAGON }
```

This is the entire contents of the file Creature.java.

l. What do you think will be produced when this file compiles successfully?

m. Notice that there is no visibility modifier on the enum types. What do you think the visibility is?

n. How many Creature objects are there?

o. Can you make any more Creature objects by using this class? Why / why not? Hint, what do you need to make an object?

### Using the extension

p. Look at the code for the CreatureDemo.java which follows.

```
/**
   This program demonstrates an enumerated type.
*/

public class CreatureDemo
{
   public static void main(String[] args)
   {
      // Declare a Creature variable and assign it a value.
     Creature good;
     Creature bad;
     Creature small;

     good = Creature.HOBBIT;
     bad = Creature.DRAGON;
     small = Creature.ELF;

      // The following statement displays HOBBIT.
      System.out.println("I am a(n) " + good);

     // "Built in" methods

     System.out.println("I am a(n) " + bad.name());

     System.out.println("Is " + good + " the same as " +
      bad + "? " + good.equals(bad) );

     if(good.compareTo(small) > 0)
      System.out.println(good + " is better than " + small + ".");
     else if (good.compareTo(small) < 0)
      System.out.println(small + " is better than " + good + ".");
     else
      System.out.println(small + " and " + good + " are the same.");

     System.out.println("The position corresponding to bad is " +
       bad.ordinal());

     System.out.println("I will now display all of the Creature objects.");
     for (Creature c : Creature.values())
      System.out.println("\t" + c);
   }
}
```

q. What value is returned by bad.name()?

2. Looking at the example on page 584, why can we use an enum type in a switch statement? Think about the kinds of values we can use in a switch and then compare how enum types are similar to those values.

3. What do we know about classes?

   a. Classes can have _____ and _____.

   b. If we say that enumerated types are a special form of a class, it implies that these classes (and their objects have _____ and _____.

c. Give an example of a method in the enum type, CarType. (Look at Day for some examples).

d. Enum types share several methods regardless of the type. They include ordinal(), name(), toString(), values(). Would it be useful to define our own methods for an enum type? Why or why not?

4. Enum types can have some "intelligence" built into them. We can define data beyond just the name of the type. Look at the Planets example from the Java Tutorial which you will find on the last page.

   a. How many Planet objects are defined by this class? _____

   b. What are they? _____

   c. What data do these objects contain? _____

   d. Does this class contain a constructor (think about what defines a constructor).

   e. Can this constructor be called by outside classes to build their own Planet? Why / Why not?

   f. Which Planet has the lowest value? The highest?

   g. What methods does the Planet class offer to users of the class?

   h. In words, describe what the main method does?

   i. Specifically, what do you think the role of the `for` loop is? Think about what the enhanced for loop does?

### Thinking ahead to design

5. How might we apply enumerated types to our own work?

   a. In what general circumstances would enumerated types have some advantages over other kinds of classes we have studied.

   b. Think about our work in 139 (or your own equivalent) and the work thus far in 239. Is there any application that might have been able to use an enumerated type to limit the set of values with which we could work?

   c. You have been asked to write an game application that will use normal playing cards. Thinking about the attributes of a playing card, what attributes would a PlayingCard class have?

   d. Would those attributes be served by an enumerated type? What might each of those look like? In other words, what would you call the type and what would be the enumerated values in each?

   e. Would these types have any methods beyond the standard ones? What are they?

```
public enum Planet
{
    MERCURY (3.303e+23, 2.4397e6),
    VENUS   (4.869e+24, 6.0518e6),
    EARTH   (5.976e+24, 6.37814e6),
    MARS    (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27,   7.1492e7),
    SATURN  (5.688e+26, 6.0268e7),
    URANUS  (8.686e+25, 2.5559e7),
    NEPTUNE (1.024e+26, 2.4746e7);

    private final double mass;   // in kilograms
    private final double radius; // in meters

    Planet(double mass, double radius)
   {
        this.mass = mass;
        this.radius = radius;
    }
    private double mass()   { return mass; }
    private double radius() { return radius; }

    // universal gravitational constant  (m3 kg-1 s-2)
    public static final double G = 6.67300E-11;

    double surfaceGravity()
    {
        return G * mass / (radius * radius);
    }
    double surfaceWeight(double otherMass)
    {
        return otherMass * surfaceGravity();
    }
    public static void main(String[] args)
    {
        double earthWeight = Double.parseDouble(args[0]);
        double mass = earthWeight/EARTH.surfaceGravity();
        for (Planet p : Planet.values())
           System.out.printf("Your weight on %s is %f%n",
                             p, p.surfaceWeight(mass));
    }
}
```