Unit Two Lecture Two:

Topic 1: Parametric Constructors continued...

Like a default constructor, the parametric constructor always has the same name as the class and is generally declared as public to enable any client code to construct new objects of the class. It must also initialize all instance fields.

Unlike a default constructor, the parametric constructor has arguments sent to it through a parameter list. In Java, these arguments are passed by "**value**" if the arguments are a primitive data type or a String. This means the address of the variable from the calling block is stored in a different location than the variable in the constructor itself. In other words, changes made to the variable in the constructor will not change the values of the variables in the calling block. However, if the variable is a class object it is passed by "**reference**". This means that changes made to the variable in the constructor will also change the values of the variables in the calling block. Therefore, it is wise not to manipulate class objects within the body of the constructor, but instead assign their values to new local variables and manipulate them.

The arguments sent to the parametric constructor are not necessarily the values that will directly initialize the instance fields. It is possible that they will be manipulated in some way to do the initialization.

Below you will find a class containing both kinds of constructors.

```
public class MyClass
{
      private int value;

      public MyClass()              // default constructor
      {
            value = 0;
      }

      public MyClass(int x, int y)  // parametric constructor
      {
            value = x + y;
      }
      .
      .
      .
}
```

```
public class Example
{
      public static void main(String[] args)
      {
            MyClass a = new MyClass();          // invokes default
                                                // constructor

            MyClass b = new MyClass(5, 7);      // invokes parametric
                                                // constructor
            .
            .
            .
      }
}
```

In Java, a class can have a default constructor, parametric constructor(s), or both.

Topic 2: Console Input

In Java version 1.5.0, a *Scanner* class was added to the Java.util package. It lets you read keyboard input in a convenient manner.

The following program shows you how the Scanner class works.

```
import java.util.Scanner;

public class Demo
{
     public static void main(String [] args)
     {
          Scanner in = new Scanner(System.in);

          int x;
          String y;
          String z;
          double r;

          System.out.print("Enter an integer value: ");
          x = in.nextInt();
          System.out.println("The value is " + x);

          System.out.print("Enter male or female: ");
          y = in.next();
          System.out.println("Gender = " + y);


          System.out.print("Enter first and last name: ");
          z = in.nextLine();
          System.out.println("Name = " + z);
```

```
            System.out.print("Enter hourly wage rate: ");
            r = in.nextDouble();
            System.out.println("Wage Rate = " + r);
      }
}
```

Topic 3: Rounding Errors/Numerical Errors

In a computer, all values are converted to binary and then all specified operations are performed on these binary numbers. The solutions are then converted back to base 10 and sent to the output. This is not a problem when all your values are integers (they all convert accurately to binary). However, rounding errors can occur when your values are doubles (floating point numbers). This happens because in binary, there is no exact representation for fractions like 1/10, just as there is no exact representation for the fraction 1/3 = .333333333 in the decimal system (base 10). When a value cannot be converted exactly, it is rounded to the nearest match. For example:

```
double f = 4.35;
System.out.println(100 * f); // Prints 434.99999999999994
```

To avoid rounding errors you can limit yourself to just using integers, as I'm going to ask you to do in U2A2, or you can use format Strings as you did with printf in U2A1.

Assignment U2A2: The Cashier Problem