Unit Four Lecture Two:

Topic 1: The do Loop

It is similar to the *while* loop, except the condition is checked in the loop terminator statement instead of the loop header statement.

It is important to note, a *do* loop will always have at least one iteration. It is possible a *while* loop will have no iterations, if the condition is false immediately.

The do loop takes the general form:

```
do
{
     statement sequence;   // body of the loop
}
while (condition(s));
```

Topic 2: The break Statement

The break statement transfers control of the program to the statement following the body of the loop in which the program was executing. In other words, the break statement allows middle exiting of loops.

Example:

```
double num;
double accum = 0;
String input;

do
{
      input = JOptionPane.showInputDialog("Enter Double: ");
      num = Double.parseDouble(input);
      if (num == -1)
            break;

      accum += num;
}
while (accum <= 100);
```

The loop will terminate as soon as a -1 is entered or as soon as the accumulator exceeds 100.

Topic 3: The continue Statement

The continue statement transfers control of the program immediately to the top of the loop for possibly another iteration. In other words, it causes the rest of the steps at the bottom of the loop to be skipped.

Example:

```
int num;
int accum = 0;
String input;

do
{
      input = JOptionPane.showInputDialog("Enter Integer: ");
      num = Integer.parseInt(input);

      if (num % 2 != 0)
            continue;

      accum += num;
}
while (accum <= 100);
```

This loop accumulates only even numbers and will terminate when the accumulator exceeds 100.

Topic 4: The flag Controlled Loop

Example:

```
int num;
boolean flag;
String input;

do
{
      flag = true;

      input = JoptionPane.showInputDialog("Enter Number: ");
      num = Integer.parseInt(input);

      if (num < 10 || num > 99)
            flag = false;
}
while (!flag);
```

This loop forces the user to enter a 2-digit number.

Topic 5: The for Loop

These loops are most often count controlled, however, they may also contain a predefined terminating condition independent of the counter.

The *for* loop takes the general form:

```
for (initialization; terminating condition; increment)
{
     statement sequence;     // body of the loop
}
```

Example 1:

```
int i;
int accum = 0;

for (i=0; i<5; i++)
{
      accum += i;
}
```

What are the values of the variables when we drop out of the loop?

i = _____   accum = _____

Example 2:

```
int c = 0;
int z;

for (int x=100; x>65; x-=5)
{
      z = Math.pow(x,2);
      c++;
}
```

What are the values of the variables when we drop out of the loop?

c = _____   z = _____   x = _____

Notice, in the two examples above the loop counter variable was declared in two different places. In example 1, the scope of i is the entire block where the loop is located. In example 2, the scope of x is only the loop itself.

Assignment U4A2: Approximating Pi

Topic 6: Random Numbers

You must use the command Math.random(). Math.random() produces random doubles ≥ 0 and < 1. Therefore, you must use an algorithm to convert these doubles into random integers you can use. This algorithm is listed below...

```
int x;

x = (int)((b-a+1) * Math.random() + a);
```

This algorithm will produce random integers between a & b inclusive.

For example: To generate random dice tosses (1 - 6), you would use...

```
x = (int)(6 * Math.random() + 1);
```

Topic 7: More Graphics

```
1) import java.awt.Graphics2D;
   import java.awt.geom.Ellipse2D;
```

Both of these classes are required if you intend to use an ellipse that can check if a specific point is contained in it.

```
2) public void paint(Graphics g)
   {
       .
       .
       .
       Graphics2D g2 = (Graphics2D) g;
       .
       .
       .
   }
```

Since the paint method requires a Graphics object as its parameter and the Ellipse2D requires a Graphics2D object, g2 is declared as a Graphics2D object and it references g which is cast as a Graphics2D object. At this point all methods of either the Graphics class or the Graphics2D class should be called through g2.

```
3) Ellipse2D.Double ellipse = new Ellipse2D.Double(1,2,10,20);
   g2.draw(ellipse);
```

The first line is declaring and instantiating a new Ellipse2D.Double object called ellipse. This ellipse will be drawn inside an invisible rectangle whose upper left corner is 1 pixel to the right and 2 pixels down from the upper left corner of the graphics window. This ellipse will have a horizontal axis of 10 pixels and a vertical axis of 20 pixels. If this ellipse is to be a circle the last 2 parameters should be equal. The second line actually draws the ellipse on the Graphics2D object.

4) `boolean inThere = ellipse.contains(x,y);`

This line returns true to the boolean variable inThere if the point x,y lies in the interior of the ellipse or on its boundary. Otherwise, it returns false.

5) `g2.fillOval(1,2,10,20);`

This line fills in the interior of the oval with the active color. This oval will be drawn inside an invisible rectangle whose upper left corner is 1 pixel to the right and 2 pixels down from the upper left corner of the graphics window. This oval will have a horizontal axis of 10 pixels and a vertical axis of 20 pixels. This method is from the Graphics class. There is a similar method in the Graphics2D class, but it requires you to instantiate a Graphics2D object before you try to fill it.

```
Ellipse2D.Double ellipse = new Ellipse2D.Double(1,2,10,20);
g2.fill(ellipse);
```

Assignment U4A2: Approximating Pi