Exam 2 Practice #1

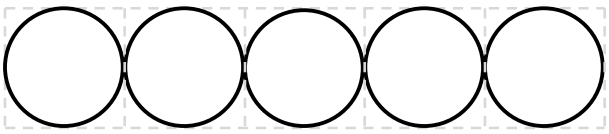
1) Assume that final variables NUM_ENTRIES, MIN_GOOD and MAX_GOOD have been declared and initialized. We'll call a number **good** if it is between MIN_GOOD and MAX_GOOD (inclusive). NUM_ENTRIES will be the number of doubles that the user will enter. Write a code segment that reads in the doubles entered by the user and computes and prints the following three numbers:
   - how many of the entries are good
   - the total (sum) of the **good** entries
   - the average of **all** of the entries

2) Write two code segments, one to generate each of the following sequences. (Hint: use our START and STEP_SIZE approach! It's OK to use magic numbers for this exercise.)
   - 39 49 59 69 79 89 99 109
   - (20, 100) (25, 98) (30, 96) (35, 94) (40, 92) (45, 90)

3) Give the exact output:

```
for (int i = 1;  i < 6;  i++)
{
    for (int j = i;  j > 0;  j--)
    {
        if ((i+ j) % 2 = = 0)
        {
            System.out.print("+");
        }
        else
        {
            System.out.print("-");
        }
        System.out.print(j);
    }
    System.out.println("");
}
```

4) Assume that the integer variables **radius** and **numberOfCircles** are class scope integers that have already been assigned values. Assume that **START_X** and **START_Y** are finals that indicate the location of the upper left corner of the first circle and that they also have already been assigned values. Write the paint method to draw:

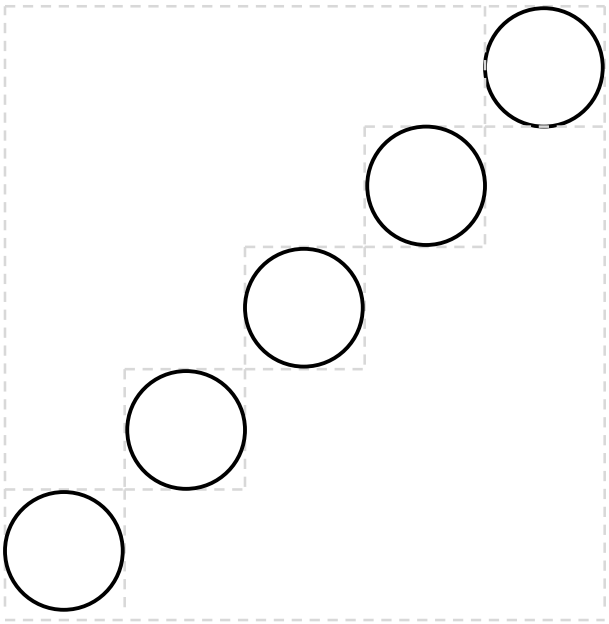


If you finish those, here are two more:

5) Write a code segment that reads in NUM_ENTRIES integers entered by the user. (Assume that the final variable NUM_ENTRIES has already been declared and initialized.) After reading in the user entries, the program should print:
   - how many entries were positive
   - how many entries were negative
   - how many entries were zero
   - which of the three categories had the most entries
     (Sample output: "Most of the numbers were negative.")

6) Assume that the variable **SIZE** is an integer final describing the width and height of the entire large square shown below and that the variable **numberOfCircles** is a class scope integer that has already been assigned a value. Assume that **START_X** and **START_Y** are finals that indicate the location of the upper left corner of the big square containing all of the circles and that they also have already been assigned values. Write the paint method to draw:

   [NOTE: The image below sometimes gets badly messed up when this file is downloaded! It should be a single diagonal row of circles, each over to the right by one diameter and up by one diameter from the previous circle.]



Solutions

1)
```
int countGood = 0;
double totalGood = 0;
double totalAll = 0;
for (int i = 1;  i <= NUM_ENTRIES; i++)
{
    System.out.println("Enter a double from " + MIN_GOOD + " to " + MAX_GOOD);
    double entry = console.nextDouble( );
    if (entry >= MIN_GOOD && entry <= MAX_GOOD)
    {
        countGood++;
        totalGood = totalGood + entry;
    }
    totalAll = totalAll + entry;
}
double averageAll = totalAll / NUM_ENTRIES;
System.out.println("There were " + countGood + " good entries.");
System.out.println("The total of the good entries was " + totalGood);
System.out.println ("The average of all of the entries was " + averageAll);
```

2) a)
```
// Here START is 39 and STEP_SIZE is 10
for (int i = 0;  i <= 7; i++)
{
    int number = 39 + i * 10;
    System.out.println(number + "  ");
}
```

b)
```
// For x, START is 20 and STEP_SIZE is 5
// For y, START is 100 and STEP_SIZE is -2
for (int i = 0;  i <= 5; i++)
{
    int x = 20 + i * 5;
    int y = 100 – i * 2;
    System.out.println( "(" + x + ", " + y + ")  " );
}
```

3)
```
+1
+2-1
+3-2+1
+4-3+2-1
+5-4+3-2+1
```

4)
```
for (int count = 0; count < numberOfCircles; count ++)
{
    int diameter = 2 * radius;
    int x = X_START + count * diameter;
    g.drawOval(x, Y_START, diameter, diameter);          // Be sure to use diameter, not radius!
}
```

5)

```
int countPos = 0;
int countNeg = 0;
int countZero = 0;
for (int i = 1;  i <= NUM_ENTRIES; i++)
{
    System.out.print (“Enter an integer: “);
    int entry = console.nextInt( );
    if (entry > 0)
    {
        countPos++;
    }
    else if (entry < 0)
    {
        countNeg++;
    }
    else
    {
        countZero++;
    }
}

// Print how many of each.
System.out.println(“There were “ + countPos + “ positive entries.”);
System.out.println(“There were “ + countNeg + “ negative entries.”);
System.out.println(“There were “ + countZero + “ zero entries.”);

// Now see which there were most of.
if (countPos >= countNeg && countPos >= countZero)
{
    System.out.println(“Most of the numbers were positive.”);
}
else if (countNeg >= countPos && countNeg >= countZero)
{
    System.out.println(“Most of the numbers were negative.”);
}
else
{
    System.out.println(“Most of the numbers were zero.”);
}
```

6)

```
int diameter = SIZE / numberOfCircles;
for (int count = 0; count < numberOfCircles; count ++)
{
    // Note that circle 0 starts at x = X_START, y = Y_START + (SIZE – diameter)
    // We could rewrite y below as y = Y_START + SIZE – (count + 1) * diameter
    int x = X_START + count * diameter;
    int y = Y_START + (SIZE – diameter) – count * diameter;

    // Be sure to use diameter, not radius!
    g.drawOval(x, y, diameter, diameter);
}
```