**Lab 13 – Hailstone Series**

Open BlueJ, and create a new BlueJ project titled **Lab13-HailstoneSeries** in your CS\LABS folder.

Create a new class with this code skeleton:

//Name:

import java.util.\*;

public class PracticeProblems

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

}

}

A while loop is very similar to an if statement, except that the code inside the code block (between the curly brackets) **will continue to run** *while* the test condition is true. Example syntax:

int i = 0;

while (i < 50) //will execute while i is less than 50

{

System.out.print(i);

i = i + 1; //can also be written as i++ or i += 1

}

Much more information can be found in our notes.

**Before each problem, insert a COMMENT with the problem number.**

1. Ask the user to enter an integer. Print the word “salsa” that many times.

Enter an integer >>> **5**

salsa salsa salsa salsa salsa

1. Ask the user to enter a new integer. Print a salsa square of that size.

Enter a new integer >>> **3**

salsa salsa salsa

salsa salsa salsa

salsa salsa salsa

1. Ask the user “Want to see this message again? (true or false)” and get a Boolean value from the keyboard. Repeat the question as long as they request.

Want to see this message again? (true or false) >>> **true**

Want to see this message again? (true or false) >>> **true**

Want to see this message again? (true or false) >>> **false**

Very well. I was getting bored with this anyway.

1. Create a loop that will add integers taken from the keyboard, except that it may add any number of integers consecutively. However, if the user does not enter at least one integer, it **should not say that the sum is 0**. Instead, it should say no integers were entered. Sample run below (**user input shown in red**):

EXAMPLE #1

Enter first integer (or 0 to quit) >>> **0**

No integers entered. Thanks for nothing, skrub.

EXAMPLE #2

Enter first integer (or 0 to quit) >>> **-7**

Enter another integer (or 0 to quit) >>> **4**

Enter another integer (or 0 to quit) >>> **1**

Enter another integer (or 0 to quit) >>> **0**

Sum of integers >>> -2

1. (Riddle) Jo's mother, Anne, is four times as old as Jo. In four years time, Anne will be three times as old as Jo. How old is Jo?
2. As a programmer at a pharmaceutical company, you’ve been tasked with writing a program that will determine how long a particular drug may stay on shelves to be sold in stores. Assume this drug loses 7% of its effectiveness every month (save into a variable so it can be easily changed for other drugs), and must be discarded when effectiveness dips below 50%. Write a program that prints how many months the drug can remain on shelves. Sample run:

Month 0: Potency: 100.0

Month 1: Potency: 93.0

Month 2: Potency: 86.49

Month 3: Potency: 80.43569999999998

Month 4: Potency: 74.80520099999998

Month 5: Potency: 69.56883692999997

Month 6: Potency: 64.69901834489997

Month 7: Potency: 60.17008706075697

Month 8: Potency: 55.95818096650398

Month 9: Potency: 52.0411082988487

Month 10: Potency: 48.39823071792929 <<< DISCARD

**Hailstone Series app**

Create a new class with this code skeleton:

//Name:

import java.util.\*;

public class HailstoneSeries

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

}

}

Write a program that asks the user for an integer variable n, from 1 to 1000. Your program should then calculate the *hailstone* *series* for that number. The hailstone series is an interesting sequence of numbers, and can be calculated like this:

* If *n* is even, the **next** number in the sequence is n/2(use integer division)
* If *n* is odd, the **next** number in the sequence is 3\*n+1

This process is repeated until you reach a value of 1. It is theorized that every integer will eventually end at 1 - this has been tested to be true for numbers up to ~1048, but it has never been proven!

Your program should print the sequence **AND the number of steps it took to reach 1**. Sample program run below (**user input shown in red**):

Enter an integer from 1 to 1000 >>> **13**

13

40

20

10

5

16

8

4

2

1

Series took 9 steps to reach a value of 1

**(Advanced) Find the maximum**

Copy your Hailstone code into a second class called HailstoneAdvanced. Add code for finding the Hailstone series such that it stores the value of the **largest integer generated by the sequence**. For the example above, the maximum would be 40. Test with some larger numbers!