Lesson 6-1: While Loops

Computer Science 46A: Introduction to Programming

San José State University

Announcements

- Homework #7 will be posted
- Lab #6 is on Friday (3/7)

Learning Objectives

By the end of this lesson, you should be able to:

1. Use a `while` loop to carry out many similar commands in succession
2. Escape from a never-ending `while` loop
3. Use a fail-safe to avoid never-ending `while` loops

Loops

- A loop is a code block that that is repeated many times
- Its one of the most powerful components of programming
- There are a few implementations of loops in different languages
- In Java, there are three different implementations of loops:
  - While loop – if a condition is met, keep doing something
  - For loop – do something a certain amount of times
  - Do loop – do something, and then keep doing it if a condition is met

The `while` loop

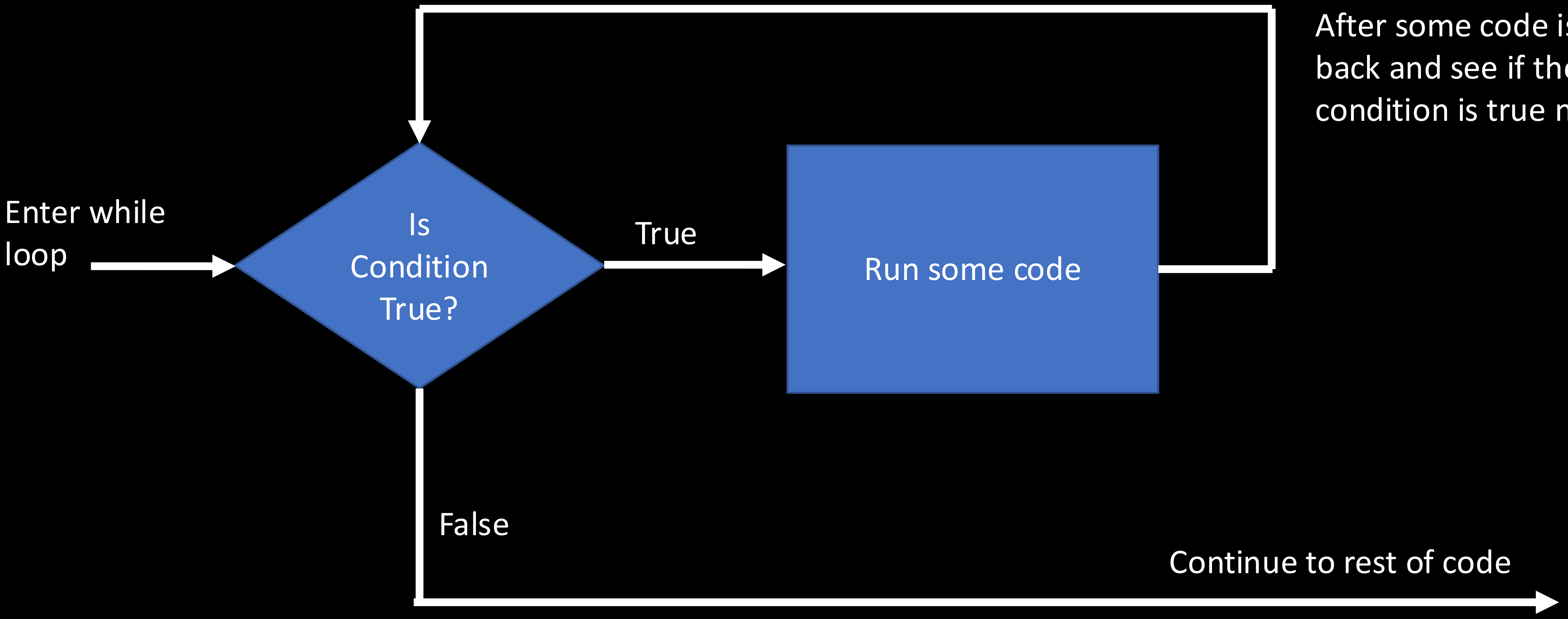
- A “while loop” is a code block that repeats a command *while* a particular condition is true
- Syntax:

```
while (condition)
{
    // do this command
}
```

as long as this is true, do this

While loops are nice tools when you do not know when something will occur

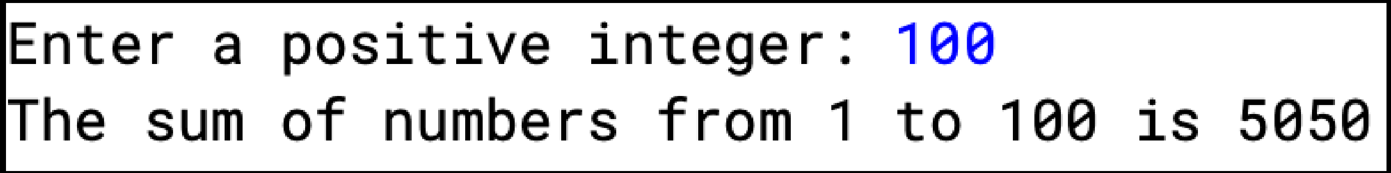
The `while` loop flow chart



Example: CountNumbers

What if we wanted to count all of the numbers between 1 and 100?

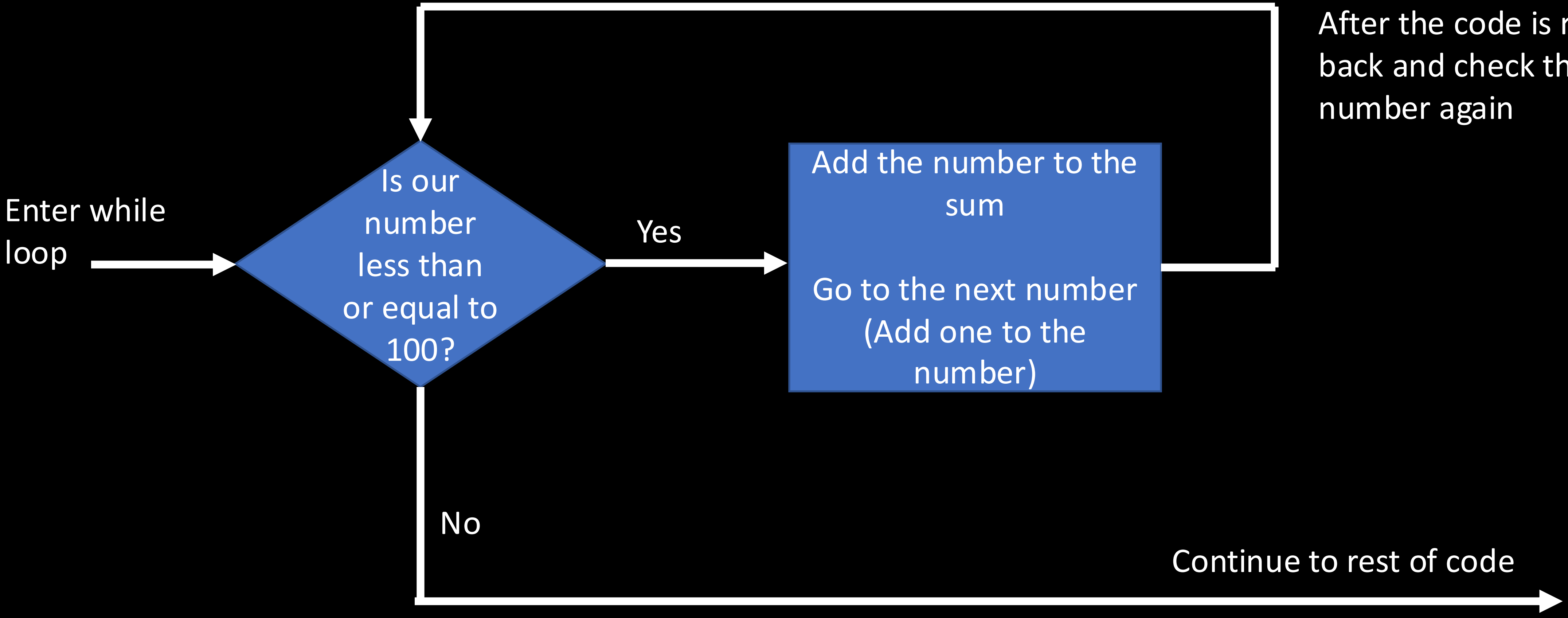
```
int number = 1;
int sum = 0;
while (number <= 100)
{
    sum = sum + number;
    number ++;
}
```



```
Enter a positive integer: 100
The sum of numbers from 1 to 100 is 5050
```

Output of CountNumbers

The `CountNumbers` flow chart


After the code is run, go back and check the number again
Enter while loop
Is our number less than or equal to 100?
Yes
Add the number to the sum
Go to the next number (Add one to the number)
No
Continue to rest of code

Poll Everywhere

Which of the following is NOT an appropriate condition for a while loop?

A. `(number>10)` where `number` is of type double

B. `(sum+1)` where `sum` is of type int

C. `(!charFound)` where `charFound` is of type boolean

D. `(len == 15)` where `len` is of type int

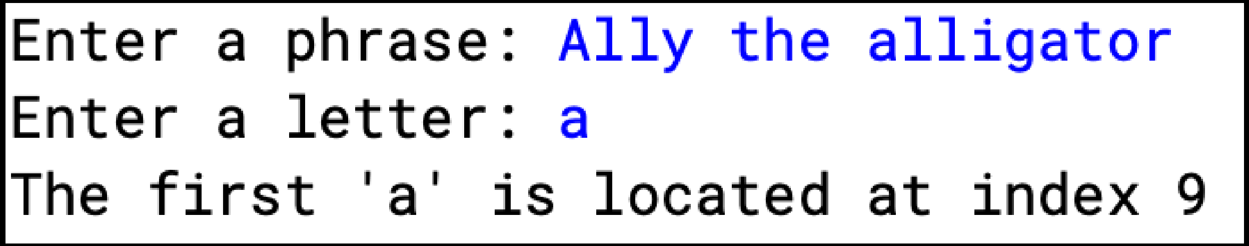
The `while` loop with other statement blocks

- We can unleash the power of loops by implementing code blocks within them
- For example, we we can implement an if/else statement into the code block:

```
while (condition)
{
    if (another condition)
    {
        // do this command
    }
}
```

Example: `FirstOccurance`

What if we wanted to find the first occurrence of the lower case letter ‘a’ in the phrase “Ally the alligator”?


```
Enter a phrase: Ally the alligator
Enter a letter: a
The first 'a' is located at index 9
```

Output of `FirstOccurance`

The Infinite Loop

- What happens if our while condition is never met?!
- For example:

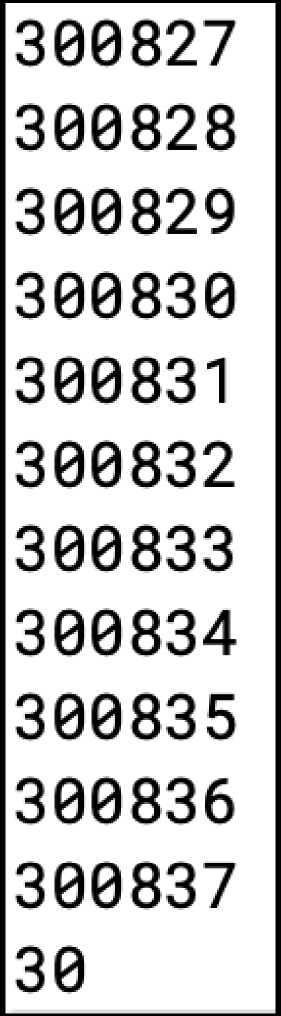
```
int i=1;
while (i>0)
{
    System.out.print(i);
    i++;
}
```

In any programming language you use,
<u>always</u> know how to escape an infinite loop!

Example: `InfiniteLoop`

What if we wanted to print a bunch of numbers in succession?

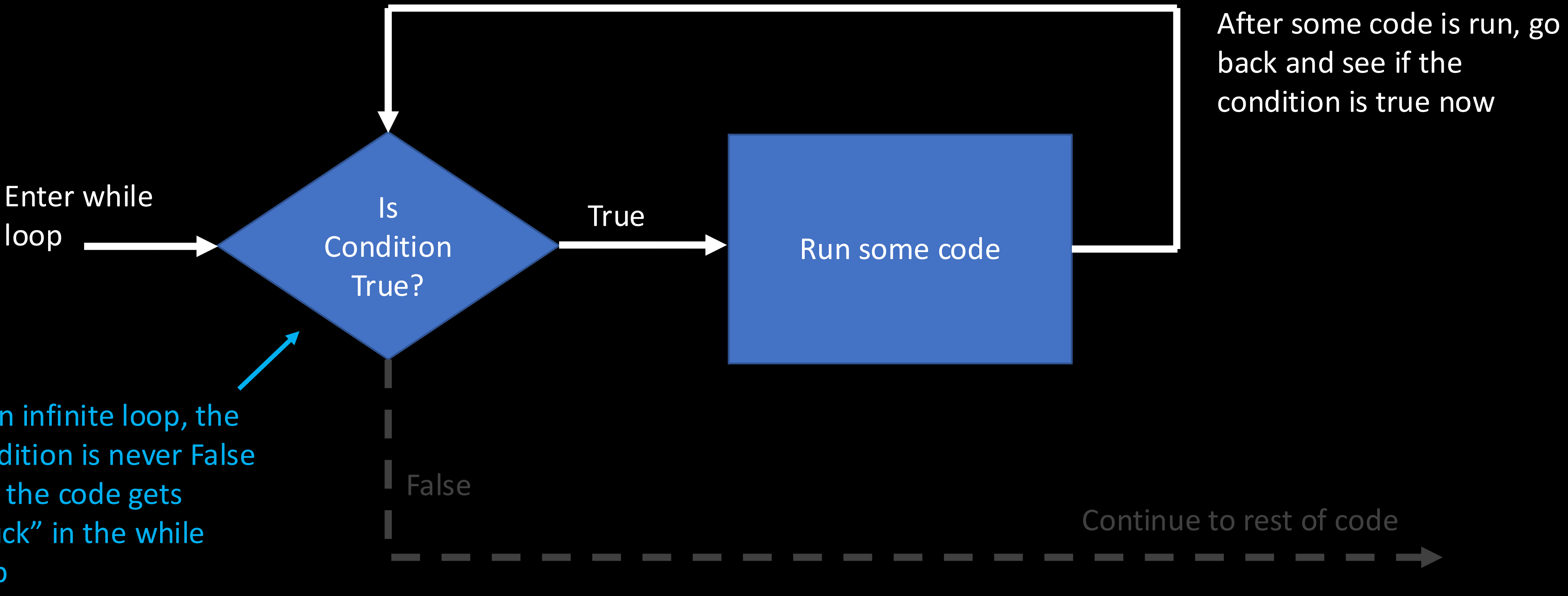
Snapshot of output from `InfiniteLoop`



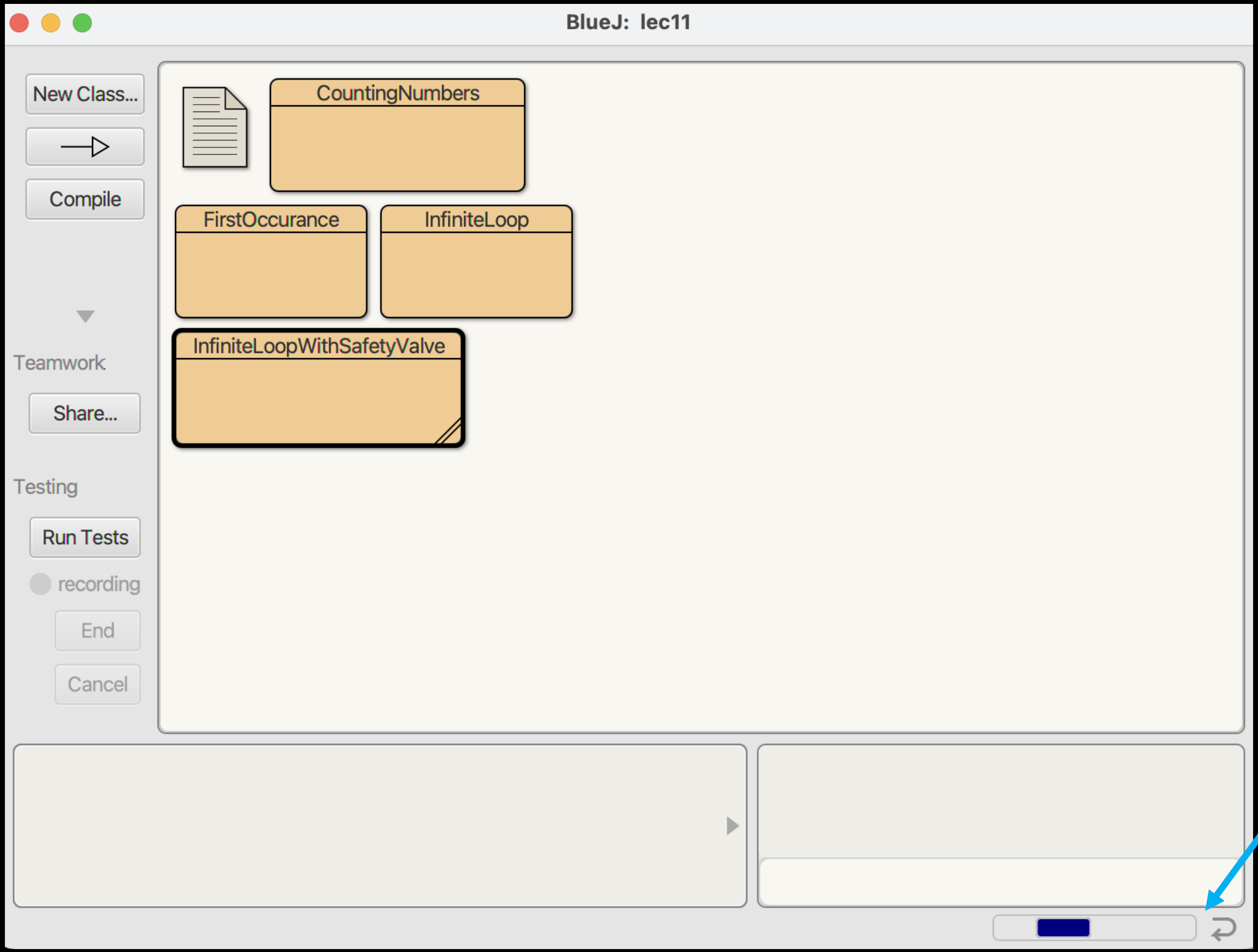
```
300827
300828
300829
300830
300831
300832
300833
300834
300835
300836
300837
30
```

An infinite loop will go forever!

The infinite `while` loop flow chart


Enter while loop
Is Condition True?
True
Run some code
After some code is run, go back and see if the condition is true now
In an infinite loop, the condition is never False and the code gets "stuck" in the while loop
False
Continue to rest of code

Escaping an infinite loop in BlueJ



When a program is running, this blue bar will be active

Right click on this bar and choose "Reset Java Virtual Machine" to terminate your program

Poll Everywhere

Which of these is NOT an infinite loop?

A)
```
boolean aFound = false;
String phrase = “hello”;
int loc = 0;
int len = phrase.length()
while (!aFound)
{
    if (phrase.charAt(loc%len)=‘a’)
    {
        aFound = true;
    }
    loc++;
}
```

B)
```
int i=0;
while (i<0)
{
    System.out.print(i);
    i++;
}
```

C)
```
int j=1;
int i=1;
while (i>0)
{
    System.out.print(i);
    j++;
}
```

PollEv.com/narayanbalasubramanian644

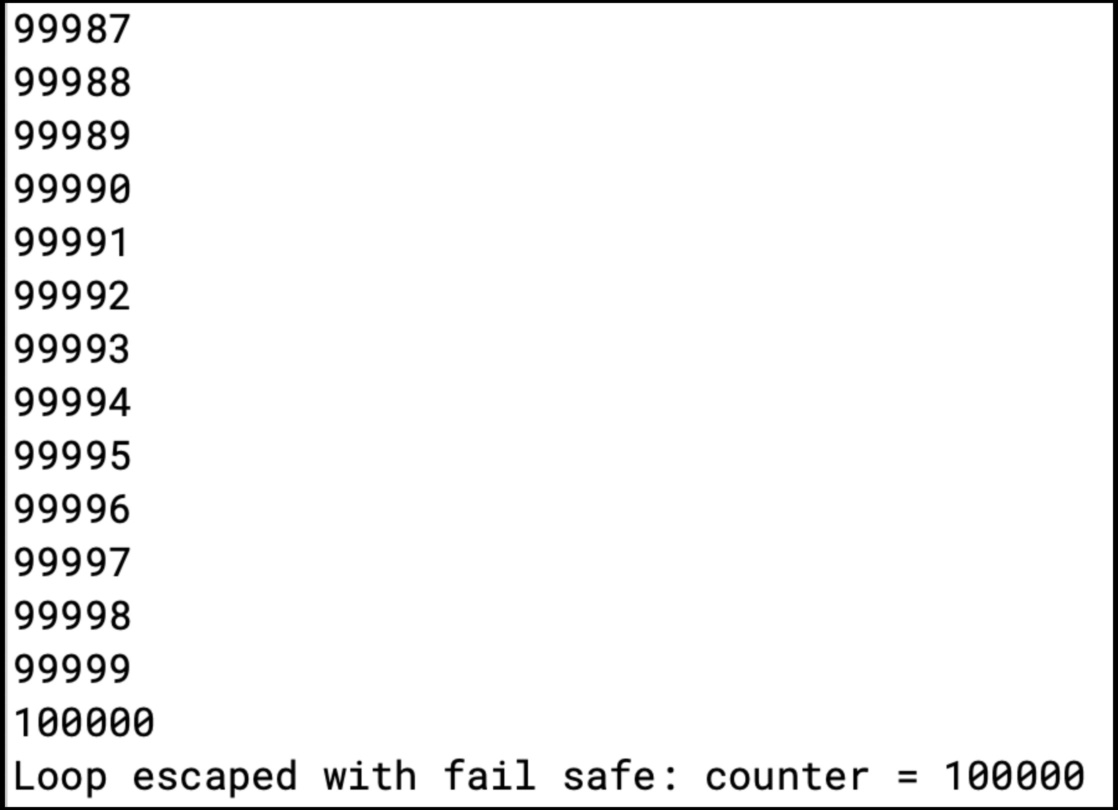
The infinite loop safety valve

- To avoid infinite loops, it can be helpful to implement a way to ensure that the loop will not go on forever
- One solution: implement a counter and a limit
  - At each iteration, increment the counter
  - In the condition for the while loop, use a *boolean operator* to include a check that the counter has not exceed the limit
  - Be sure that you limit does not impede the purpose of your loop!

Example: `InfiniteLoopWithSafetyValve`

What if we wanted to print a bunch of numbers in succession?

But not forever!

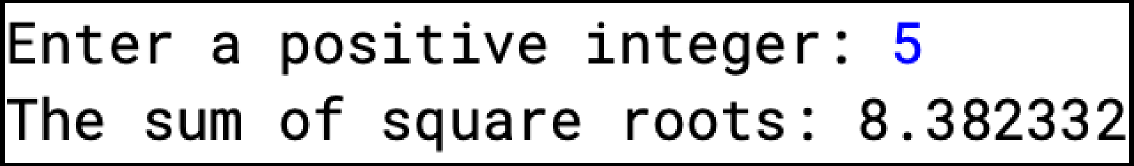


```
99987
99988
99989
99990
99991
99992
99993
99994
99995
99996
99997
99998
99999
100000
Loop escaped with fail safe: counter = 100000
```

End of output from `InfiniteLoopWithSafetyValue`

Participation Exercise 6-1a: SumOfSqrt

Goal: Calculate the sum of the *square roots* of all numbers between 1 and a given number



```
Enter a positive integer: 5
The sum of square roots: 8.382332
```

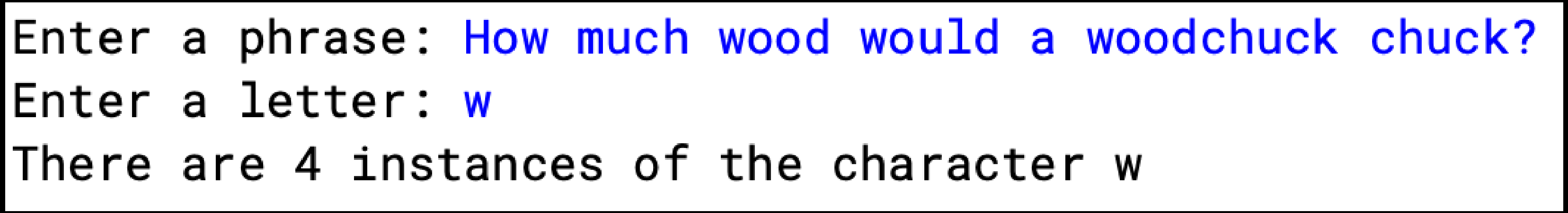
Output of SumOfSqrt

Hint: Use the Math.sqrt() method of the Math class

Codecheck Link: HERE and on Canvas

Participation Exercise 6-1b: `CountOccurrences`

Goal: Count the number of occurrences of a letter in a given phrase



```
Enter a phrase: How much wood would a woodchuck chuck?
Enter a letter: w
There are 4 instances of the character w
```

Example output of `CountOccurrences`

Codecheck Link: HERE and on Canvas