https://runestone.academy/runestone/books/published/apcsareview/LoopBasics/lbasics.html

Answers are in Bold

7.1 Loops in Java

There are 3 different types of loops in Java. The While loop, which runs until a statement or condition has been met. The For loop which has 3 parts in its header, the declaration/initialization, the condition, and then some form of change. This repeats the code block until the condition is true. The change code is executed at each run-through of the loop. And lastly the For-Each loop, which loops through data and each time sets a variable equal to the item it is iterated upon in the data.

Activity 1 While Loop Example:

Activity 2 For Loop Example:

Activity 3:

Activity 4:

trl-2: Click on all the statements that are part of the body of the for loop. If you make a mistake you can click on the statement again to unhighlight it.

```
for (int x = 5; x > 0; x--)
System.out.println(x);

Check Me

You are Correct!

Activity: 4 -- Clickable (click_for1)
```

Activity 5:

trl-3: Click on all the statements that are part of the body of the for loop. If you make a mistake you can click on the statement again to unhighlight it.

Check Me

You are Correct!

Activity: 5 -- Clickable (click_for2)

7.2 While Loops

While loops are generally used when you do not know how many times something will be executed, as it will loops the code block until the conditional given is false. For example a loop such as while(true) would loop forever, as the statement is always true.

Good note for the exam is to trace out iterations of a loop using a table, as described in 7.2.1 where they show this through this example.

Click on the following link to step through the code above with the Java Visualizer - Click here.

You can create a table that keeps track of the variable values each time through the loop as shown below. This is very helpful on the exam. Studies have shown that students who create tables like this do much better on code tracing problems on multiple choice exams.

iteration	var1	vara
0	3	2
1	4	1
2	5	0

Figure 1: A table showing the values of all of the variables each time through the loop. The 0 means before the first loop.

7.2 Check your Understanding:

```
6-2-2: What are the values of var1 and var2 when the code finishes executing?

int var1 = 0;
int var2 = 2;

while ((var2 != 0) && ((var1 / var2) >= 0))
{
    var1 = var1 + 1;
    var2 = var2 -1;
}

A. var1 = 1, var2 = 1

B. var1 = 2, var2 = 0

C. var1 = 3, var2 = 1

D. var1 = 0, var2 = 2

E. The loop will cause a run-time error with a division by zero

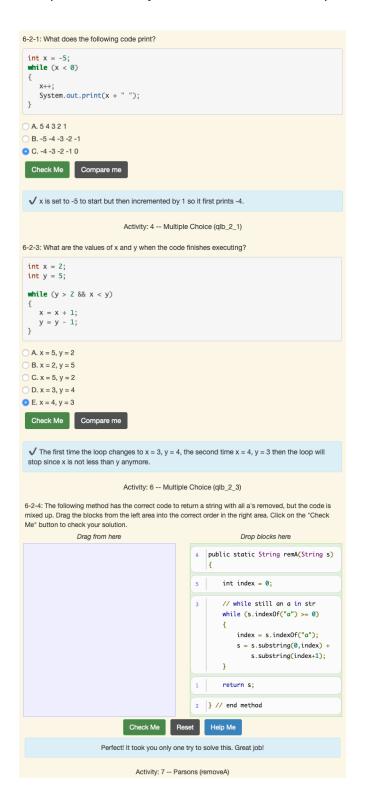
Check Me

Compare me

The loop stopped because var2 = 0. After the first execution of the loop var1 = 1 and var2 = 1.

After the second execution of the loop var1 = 2 and var2 = 0. This stops the loop and doesn't execute the second part of the complex conditional.

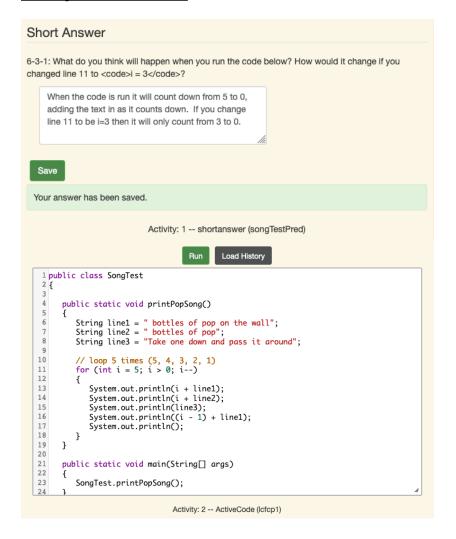
Activity: 5 -- Multiple Choice (qlb_2_2)
```



7.3 For Loops

For Loops in Java are almost identical to those in Python, but I'll write down notes nonetheless. For Loops have 3 main parts; the initialization, the condition, and the change. They're written in Java like this: for(initialization; condition; change) The Initialization is run only once, before the loop is run and is generally used to create a temporary variable. The condition uses the variable created in initialization to check for something. If the condition is still true then the loop will run. Then the change runs after the loop has run, changing the initialization in some way. You can calculate the number of iteration that will be run by subtracting the smallest number that will let the loop run from the largest number that will let the loop run, and then adding one to account for running at the initial condition. (The value that ends the loop - the starting value). Watch for = signs such as < and <=. If you have a < or > then it is 1 less than the value being checked, because the value being check is not included. But if it's <= or >= the value IS included and therefore checked.

Activity 1 Short Answer:



7.3 Check your Understanding:



7.4 Nested For Loops

Nest For Loops are just for loops placed inside other for loops. They're useful for working in multiple dimensions such as printing columns each time you print a row. You can calculate the number of times the loop runs by multiplying the number of times each loop runs by each other.

7.4 Check your Understanding:

```
6-4-1: How many times does the following code print a *?
  for (int i = 3; i < 8; i++)
      for (int y = 1; y < 5; y++)
          System.out.print("*");
      System.out.println();
O A. 40
OB. 20
O. 24
O D. 30

√ The outer loop executes 7-3+1=5 times and the inner 4-1+1=4 so this will print 5 * 4 = 20 stars.

                              Activity: 2 -- Multiple Choice (qln_6_1)
6-4-2: What does the following code print?
  for (int i = 2; i < 8; i++)
      for (int y = 1; y \le 5; y++)
          System.out.print("*");
      System.out.println();
 A. A rectangle of 8 rows with 5 stars per row.

    B. A rectangle of 8 rows with 4 stars per row.

O. A rectangle of 6 rows with 5 stars per row.

    D. A rectangle of 6 rows with 4 stars per row.

 Check Me Compare me
  ✓ The outer loop executes 8-2+1=6 times so there are 6 rows and the inner loop executes
 5-1+1=5 times so there are 5 columns.
                              Activity: 3 -- Multiple Choice (qln_6_2)
6-4-3: What does the following print?
  for (int i = 3; i \le 9; i++)
     for (int j = 6; j > 0; j--)
         System.out.print("*");
     System.out.println();

    A. A rectangle of 9 rows and 5 stars per row.

    B. A rectangle of 6 rows and 6 stars per row.

 C. A rectangle of 7 rows and 5 stars per row.
O D. A rectangle of 7 rows and 6 stars per row.
  Check Me Compare me
  ✓ The outer loop executes 9 - 3 + 1 = 7 times and the inner 6 - 1 + 1 = 6 times.
                              Activity: 4 -- Multiple Choice (qln_6_3)
```