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:

Run Original - 1 of 1

```
public class Test
{
   public static void main(String[] args)
   {
       int x = 3;
       while (x > 0)
       {
          System.out.println(x);
          x = x - 1;
       }
   }
}
```

```
3
2
1
```

Activity: 1 -- ActiveCode (while_loop_ex1)

Activity 2 For Loop Example:

Run Original - 1 of 1

```
public class Test
{
   public static void main(String[] args)
   {
       for (int x = 3; x > 0; x--)
       {
          System.out.println(x);
       }
   }
}
```

```
3
2
1
```

Activity: 2 -- ActiveCode (for_loop_ex1)

Activity 3:

trl-1: Click on all the statements that are part of the body of the while loop. If you make a mistake you can click on the statement again to unhighlight it.

```
int x = 5;
while (x > 0)
{
    System.out.println(x);
    x = x - 1;
}
```

Check Me

You are Correct!

Activity: 3 -- Clickable (click_while1)

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.

```
String message1 = "I ";
String message2a = "love ";
String message3 = "you";
String message2b = "miss ";
for (int x = 1; x < 4; x++)
{
    System.out.println(message1 + message2a + message3);
    System.out.println(message1 + message2b + message3);
}
```

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.

Run Original - 1 of 1

```
public class Test
{
    public static void main(String[] args)
    {
        int var1 = 3;
        int var2 = 2;

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

Activity: 2 -- ActiveCode (example_trace_loop)

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	var2
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)

6-2-1: What does the following code print?

```
int x = -5;
while (x < 0)
{
   x++;
   System.out.print(x + " ");
}
```

- A. 5 4 3 2 1
- B. -5 -4 -3 -2 -1
- C. -4 -3 -2 -1 0

Check Me | Compare me

✓ x is set to -5 to start but then incremented by 1 so it first prints -4.

Activity: 4 -- Multiple Choice (qlb_2_1)

6-2-3: What are the values of x and y when the code finishes executing?

```
int x = 2;
int y = 5;

while (y > 2 && x < y)
{
   x = x + 1;
   y = y - 1;
}
```

- A. x = 5, y = 2
- B. x = 2, y = 5
- C. x = 5, y = 2
- D. x = 3, y = 4
- E. x = 4, y = 3

Check Me | Compare me

✓ The first time the loop changes to x = 3, y = 4, the second time x = 4, y = 3 then the loop will stop since x is not less than y anymore.

Activity: 6 -- Multiple Choice (qlb_2_3)

6-2-4: The following method has the correct code to return a string with all a's removed, but the code is mixed up. Drag the blocks from the left area into the correct order in the right area. Click on the "Check Me" button to check your solution.

Drag from here | *Drop blocks here*

```
4 public static String remA(String s)
  {
5     int index = 0;
3     // while still an a in str
      while (s.indexOf("a") >= 0)
      {
          index = s.indexOf("a");
          s = s.substring(0,index) +
              s.substring(index+1);
      }
1     return s;
2 } // end method
```

Check Me | Reset | Help Me

Perfect! It took you only one try to solve this. Great job!

Activity: 7 -- Parsons (removeA)

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:

Short Answer

6-3-1: What do you think will happen when you run the code below? How would it change if you changed line 11 to <code>i = 3</code>?

When the code is run it will count down from 5 to 0, adding the text in as it counts down. If you change line 11 to be i=3 then it will only count from 3 to 0.

Save

Your answer has been saved.

Activity: 1 -- shortanswer (songTestPred)

Run Load History

```
1 public class SongTest
2 {
3
4    public static void printPopSong()
5    {
6       String line1 = " bottles of pop on the wall";
7       String line2 = " bottles of pop";
8       String line3 = "Take one down and pass it around";
9
10      // loop 5 times (5, 4, 3, 2, 1)
11      for (int i = 5; i > 0; i--)
12      {
13         System.out.println(i + line1);
14         System.out.println(i + line2);
15         System.out.println(line3);
16         System.out.println((i - 1) + line1);
17         System.out.println();
18      }
19   }
20
21   public static void main(String[] args)
22   {
23      SongTest.printPopSong();
24   }
```

Activity: 2 -- ActiveCode (lcfcp1)

7.3 Check your Understanding:

6-3-2: What does the following code print?

```
for (int i = 3; i < 8; i++)
{
   System.out.print(i + " ");
}
```

- A. 3 4 5 6 7 8
- B. 0 1 2 3 4 5 6 7 8
- C. 8 8 8 8 8
- D. 3 4 5 6 7

Check Me | Compare me

✓ The value of i is set to 3 before the loop executes and the loop stops when i is equal to 8. So the last time through the loop i is equal to 7.

Activity: 4 -- Multiple Choice (qlb_3_1)

6-3-3: What does the following code print?

```
for (int i = 1; i <= 10; i++)
{
   System.out.print(i + " ");
}
```

- A. 3 4 5 6 7 8
- B. 0 1 2 3 4 5 6 7 8 9
- C. 1 2 3 4 5 6 7 8 9 10
- D. 1 3 5 7 9

Check Me | Compare me

✓ The value of i starts at 1 and this loop will execute until i equals 11. The last time through the loop the value of i is 10.

Activity: 5 -- Multiple Choice (qlb_3_2)

6-3-4: How many times does the following method print a `*` ?

```
for (int i = 3; i <= 9; i++)
{
   System.out.print("*");
}
```

- A. 10
- B. 6
- C. 7
- D. 9

Check Me | Compare me

✓ How many numbers are between 3 and 9 (including 3 and 9)?

Activity: 6 -- Multiple Choice (qlb_3_3)

6-3-5: The following method has the correct code to print out all the even values from 0 to the value of 10, but the code is mixed up. Drag the blocks from the left into the correct order on the right and indent them correctly. Even though Java doesn't require indention it is a good habit to get into. You will be told if any of the blocks are in the wrong order or not indented correctly when you click the "Check Me" button.

Drag from here | *Drop blocks here*

```
2 public static void printEvens()
  {
    1 for (int i = 0;
              i <= 10;
              i+=2)
      {
        3 System.out.println(i);
      5 } // end for
  4 } // end method
```

Check Me | Reset | Help Me

Perfect! It took you only one try to solve this. Great job!

Activity: 7 -- Parsons (print_evens)

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();
}
```

- A. 40
- B. 20
- C. 24
- D. 30

Check Me Compare me

✓ 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 <= 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.
- C. 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 <= 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.
- 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)