Unit 4: Iteration

Mr. Pelletier, Lord Byng Secondary

Unit 4: Iteration

Overview

We will be covering:

- While loops
- For loops
- Strings and Loops - Builders, Parsers, and Searchers
- Nested Iteration
- Informal Code Analysis

4.1: While Loops

Mr. Pelletier, Lord Byng Secondary

4.1 While Loops

Overview

While loops allow us to repeat a block of code as long as a particular boolean expression is true. It looks like this:

```
while (boolean expression) {

  //code statements

}
```

4.1 While Loops

The Basics 1/3

Here is an example of a while loop. What is outputted to the console?

```
int x = 0;
while (x < 10) {
    System.out.println(x);
    x++;
}
```

4.1 While Loops

Basics 2/3

This is similar to an if statement, but in a while loop, the flow of control returns to the boolean condition after the code block finishes. Compare the output of these two control structures if i = 5:

```
while (x < 10) {

    System.out.println(x);

    x++;

}
```

```
if (x < 10) {

    System.out.println(x);

    x++;

}
```

4.1 While Loops

Basics 3/3

What is the output if x = 5?

```
while (x < 10) {
    System.out.println(x);
    x++;
}
```

```
if (x < 10) {
    System.out.println(x);
    x++;
}
```

4.1 While Loops

Examples 1/4

Write a code segment that asks a user if they like AP Computer Science. The program keeps asking until you say “yes”. Then it congratulates you.

```
Scanner input = new Scanner(System.in);
System.out.print(“Do you like AP Computer Science? “);
String answer = input.nextLine();
...
```

4.1 While Loops

Examples 2/4

```
Scanner input = new Scanner(System.in);
System.out.print(“Do you like AP Computer Science? “);
String answer = input.nextLine();

while (!(“yes”.equals(answer)) {
  System.out.print(“Do you like AP Computer Science? “);
  answer = input.nextLine();
}

System.out.println(“I am so glad you like it! There is a
test next week.”);
```

4.1 While Loops

Examples 3/4

Write a while loop that takes in integers from the terminal until you enter a negative number. It then tells you the total of all the numbers entered.

```
Scanner input = new Scanner(System.in);
int total = 0; //keeps track of total
int num = 0;   //stores a single number entered by user
```

4.1 While Loops

Examples 4/4

```
Scanner input = new Scanner(System.in);
int total = 0; //keeps track of total
int num = 0;   //stores a single number entered by user

while (num >= 0) {
  num = input.nextInt();
  if (num > 0) total += num;
}

System.out.println(“Your total was: “ + total);
```

4.1 While Loops

Infinite Loops

If a loop can't end, it is called "infinite." Generally these are the result of bugs. If this happens in your code, your code may appear to "freeze" and nothing happens. If there is output in the loop, you might see it flooding the console. For example:

```
int i = 10;
while (i > 0) {
  System.out.println(i);
  i++;
}
```

4.1 While Loops

Do while (NOT ON EXAM)

Not on the exam, but sometimes useful to know is the do while loop. It checks the boolean expression at the END instead of the beginning. What is the output of this if i = 15?

```
do {

    System.out.println(x);

    x++;

} while (x < 10);
```

4.1 While Loops

break (NOT ON EXAM)

You can manually cause a loop to finish in the middle of its execution with the break keyword.

```
int i = 10;
while (true) {
  i++;
  if (i == 100)
    break;
}
```

4.2: For Loops

Mr. Pelletier, Lord Byng Secondary

4.2 For Loops

Overview

A for loop is a convenient syntax shortcut for a particular kind of loop.

```
for (initialization; boolean expression; iteration) {
  //code statements
}
```

4.2 For Loops

The Basics 1/6

Here is an example of a while converted into a for loop. They do the same thing!

```
int x = 0;
while (x < 10) {
    System.out.println(x);
    x++;
}
```

```
for (int x = 0; x < 10; x++) {
    System.out.println(x);
}
```

4.2 For Loops

The Basics 2/6

Important components of the loop have been placed up front, all in one place.

```
int x = 0;
while (x < 10) {
    System.out.println(x);
    x++;
}
```

```
for (int x = 0; x < 10; x++) {
    System.out.println(x);
}
```

4.2 For Loops

The Basics 3/6

Initialization only happens at The very start of the loop. It is not repeated

Initialization

```
int x = 0;

while (x < 10) {

    System.out.println(x);

    x++;

}
```

Initialization

```
for (int x = 0; x < 10; x++) {

    System.out.println(x);

}
```

4.2 For Loops

The Basics 4/6

In the for loop, x only has scope inside the loop.

Initialization

```
int x = 0;

while (x < 10) {

    System.out.println(x);

    x++;

}
```

In the while loop, x has scope inside and outside the loop.

Initialization

```
for (int x = 0; x < 10; x++) {

    System.out.println(x);

}
```

4.2 For Loops

The Basics 5/6

Boolean Expression

```
int x = 0;
while (x < 10) {
    System.out.println(x);
    x++;
}
```

Boolean Expression

```
for (int x = 0; x < 10; x++) {
    System.out.println(x);
}
```

4.2 For Loops

The Basics 6/6

Initializing a counting variable goes first

Note that incremenation always happens at the very end of the for loop

```
int x = 0;

while (x < 10) {

    System.out.println(x);

    x++;

}
```

Incrementation

```
for (int x = 0; x < 10; x++) {

    System.out.println(x);

}
```

Incrementation

4.2 For Loops

While Loops vs. For Loop

While loops is the most general loop. It can do anything.

For loops are optimized for repeating in predefined patterns.

Do you know how many times a loop should run? Probably use a for loop.

Do you not know how many times a loop should run? Probably use a while loop.

4.3: Developing Algorithms for Strings

Mr. Pelletier, Lord Byng Secondary

4.3 Developing Algorithms for Strings

Overview

In this lesson, we will be using loops to explore Strings. In particular, we will be creating three different kinds of algorithms:

String Builders: Loop through a given String and return a new String that is based on the given String but changed in some way. Eg: doubleChar(), mixString()

String Counter: Loop through a given String and count the number of occurrences of some pattern. Eg: countHi(), countCode()

String Searchers: Loop through a given String and return true or false based on some criteria. Eg: catDog(), xyzThere()

4.3 Developing Algorithms for Strings

The Core Loop 1/3

The main concept that connects all these algorithms is “looping through” a String.

The loop increments an int variable that is a parameter to a substring method call. As the loop proceeds, the variable moves down the string.

4.3 Developing Algorithms for Strings

The Core Loop 2/3

Example: Write a method that takes a String and returns a new String with spaces after each letter.
Eg: spaceOut(“abc”) returns “a b c ”.

What is the purpose of str? i? snippet? answer?

```
public String spaceOut(String str) {
  String answer = “”;
  for (int i = 0; i < str.length(); i++) {
    String snippet = str.substring(i, i+1);
    response += snippet + “ ”;
  }
  return answer;
}
```

4.3 Developing Algorithms for Strings

The Core Loop 3/3

```
public String spaceOut(String str) {
  String answer = “”;
  for (int i = 0; i < str.length(); i++) {
    String snippet = str.substring(i, i+1);
    answer += snippet + “ ”;
  }
  return answer;
}
```

What is the purpose of str? i? snippet? answer?

D	a	r	t	h
0	1	2	3	4

4.3 Developing Algorithms for Strings

IndexOutOfBounds 1/2

When looking at two or more characters in your snippet, be careful not to go out of bounds! For example:

```
public String countTh (String str) {
  int count = 0;
  for (int i = 0; i < str.length(); i++) {
    String snippet = str.substring(i, i+2);
    if (“th”.equals(snippet)) count++;
  }
  return count;
}
```

D	a	r	t	h
0	1	2	3	4

4.3 Developing Algorithms for Strings

IndexOutOfBounds 2/2

One way to handle this:

```
public String countTh (String str) {
  String count = 0;
  for (int i = 0; i <= str.length() - 2; i++) {
    String snippet = str.substring(i, i+2);
    if (“th”.equals(snippet)) count++;
  }
  return count;
}
```

D	a	r	t	h
0	1	2	3	4

4.4: Nested Loops

Mr. Pelletier, Lord Byng Secondary

4.4 Nested Loops

Overview

A nested loop is a loop within another loop.

You will be required to read and nested loops to determine their output, and write nested loops to create a desired output.

The key concept is that the **inner loop must fully complete** before the outer loop can continue.

4.4 Nested Loops

Example 1/10

Let’s make a loop that prints out the time in 24 hour format, simulating this pattern:

00:00 —> 00:59

01:00 —> 01:59

02:00 —> 02:59

4.4 Nested Loops

Example 2/10

We can use a nested loop. Inner loop controls minutes, outer loop controls hours.

```
for (int hrs = 0; hrs < 12; hrs++) {
  for (int min = 0; min < 60; min++) {
    System.out.println(hrs + ":" + min);
  }
}
```

4.4 Nested Loops

Example 3/10

We can use a nested loop. Inner loop controls minutes, outer loop controls hours.

```
for (int hrs = 0; hrs < 12; hrs++) {
  for (int min = 0; min < 60; min++) {
    System.out.println(hrs + ":" + min);
  }
}
```

The Initialization statement of the inner loop resets min to zero everytime the inner loop begins.

4.4 Nested Loops

Example 4/10

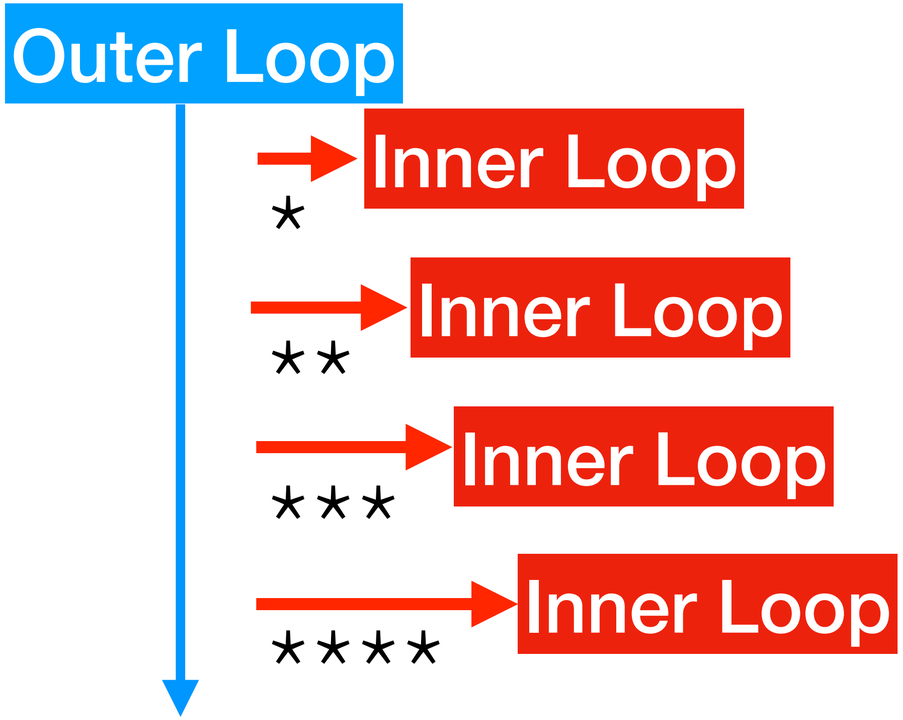
What if we wanted to print the following pattern to the console?

```
*
**
***
****
```

4.4 Nested Loops

Example 5/10

What if we wanted to print the following pattern to the console?



4.4 Nested Loops

Example 6/10

Answer:

```
for (int outer = 1; outer <= 4; outer++) {
  for (int inner = 0; inner < outer; inner++) {
    System.out.print(“*”);
  }
  System.out.println(“”);
}
```

4.4 Nested Loops

Example 7/10

What if we wanted to print the reverse of this pattern to the console?

```
****
***
**
*
```

4.4 Nested Loops

Example 8/10

Answer:

```
for (int outer = 4; outer > 0; outer++) {
  for (int inner = 0; inner < outer; inner++) {
    System.out.print(“*”);
  }
  System.out.println(“”);
}
```

4.4 Nested Loops

Example 9/10

What is the output of this loop?

```
for (int outer = 1; outer < 5; outer++) {
  for (int inner = 1; inner < 3; inner++) {
    System.out.print(inner + “ ”);
  }
  System.out.println(“”);
}
```

4.4 Nested Loops

Example 10/10

Swap these two

Now, what if we made these changes? What is the output now?

```
for (int outer = 1; outer < 5; outer++) {
  for (int inner = 1; inner < 3; inner++) {
    System.out.print(inner + “ ”);
  }
  System.out.println(“”);
}
```

4.5: Informal Code Analysis

Mr. Pelletier, Lord Byng Secondary

4.5 Informal Code Analysis

Overview

In this lesson we will count “statement execution counts,” which means how many times a particular statement runs.

We will look at common patterns and calculate statement execution counts from code examples.

4.5 Informal Code Analysis

Statement Execution Counts

```
public static void main(String[] args) {
    int count = 0;
    for(int k = 0; k < 30; k++)
    {
      if(k % 3 == 0) //line 1
      {
          count++; //line 2
      }
    }
    System.out.println(count);
  }
```

What is the statement execution count for statement one? How about statement two?

4.5 Informal Code Analysis

Statement Execution Counts

```
for(int k = 0; k < 135; k++)
{
    if (k % 5 == 0)
    {
        System.out.println(k);
    }
}
```

How can we rewrite this loop to run faster based on statement execution counts?

4.5 Informal Code Analysis

Statement Execution Counts

```
for(int k = 0; k < 135; k++)
{
    if (k % 5 == 0)
    {
        System.out.println(k);
    }
}
```

```
for (int k = 0; k < 135; k+=5)
{
    System.out.println(k);
}
```

How can we rewrite this loop to run faster based on statement execution counts?

4.5 Informal Code Analysis

Statement Execution Counts

Which statement will execute more times?

```
for(int k = 1; k < 100; k++)
{
    //statement #1
}
```

```
int count = 0;
int k = 1;
while(k < 100)
{
    //statement #2
    k++;
}
```

4.5 Informal Code Analysis

Statement Execution Counts

How many times does statement 1 execute?

```
for(int k = 0; k < 1000; k++)
{
      //statement #1
}
```

4.5 Informal Code Analysis

Statement Execution Counts

How many times does statement 1 execute?

```
for(int k = 6; k < 50; k++)
{
    //statement #1
}
```

4.5 Informal Code Analysis

Statement Execution Counts

How many times does statement 1 execute?

```
for (int outer = 0; outer < 3; outer++)
{
    for (int inner = 0; inner < 4; inner++)
    {
        //statement #1
    }
}
```

4.5 Informal Code Analysis

Statement Execution Counts

How many times does statement 1 execute?

```
for (int outer = 5; outer > 0; outer--)
{
    for (int inner = 0; inner < outer; inner++)
    {
        //statement #1
    }
}
```

4.5 Informal Code Analysis

Statement Execution Counts

```
int k = 1;
while(k <= 7)
{
  for(int z = 0; z < 4; z++)
  {
     //statement #1
  }
  k++;
}
```

How many times does statement 1 execute?

4.5 Informal Code Analysis

Statement Execution Counts

```
int k = 0;
while(k < 5)
{
  int x = (int)(Math.random()*6) + 1;
  while(x != 6)
  {
     //statement #1
     x = (int)(Math.random()*6) + 1;
  }
  k++;
}
```

How many times does statement 1 execute?

Big-O Complexity Chart