Chad Philip Johnson
CSCI20, Worthington
September 03rd, 2012

Pencil & Paper 05

1. In general terms, what is a loop?
A loop is a section of code that repeats until a certain Boolean condition becomes false. For *while* and *do-while* loops the code within the loop's body repeats until the variable or statement being tested returns *false*, or a *break* statement is encountered. For *for* loops the code within the loop's body repeats until a variable or statement being tested returns false, or a *break* statement is encountered. *for* loops also have built in initialization and update parameters.

2. What is an *infinite loop*?
An infinite loop is a *while*, *do-while* or *for* loop that never exits. According to our textbook, it is "a loop that runs forever" (144).

3. According to your text, what is the main difference between a while loop and a do-while loop?
A *while* loop may never execute, depending on the condition(s) being tested, whereas a *do-while* loop will always execute at least one time. The explanation given by our textbook is "with a *while* statement, the Boolean expression is checked before the loop body is executed…. With a *do-while* statement, the body of the loop is executed first and the Boolean expression is checked after the loop body is executed. Thus, the *do-while* statement always executes the loop body at least once. After this start-up, the *while* loop and the *do-while* loop behave the same way" (134-135).

4. What output is produced by the following Java code?

```
int counter = 1;
while (counter < 5)
{
        System.out.println("Counter = " + counter);
        counter++;
}
System.out.println("Counter at end of loop: " + counter);
```

The following is printed to the console:

Counter = 1
Counter = 2
Counter = 3
Counter = 4
Counter at end of loop: 5

5. What output is produced by the following Java code?

```
int counter = 3;
do
{
        counter--;
        System.out.println("Counter = " + counter);
} while (counter > 0);
```

The following is printed to the console:

Counter = 2

Counter = 1
Counter = 0

6. Write a while loop that displays the values 1 through 10 (counting up) on a single line, with commas in between them (no line breaks). You will need to declare a variable for this loop.

The following code performs these operations:

```
int counter = 1;
while( counter < 10 )        {
        System.out.print( counter + "," );
        counter++;
}
System.out.print( counter );
```

7. Write a for loop that displays the values 100 through 1 (counting down) on a single line, with spaces in between them (no line breaks).

The following code performs these operations:

```
for( int counter = 100; counter > 0; counter-- )  {
        System.out.print( counter + " " );
}
```

8. What is the output of the following for loop (*trick question)?

```
for (int number=0; number > 10; number++)
{
        System.out.println("counting down…");
}
```

There is no output for this loop: the condition *number > 10* is made false by the initialization statement *int number = 0*.

9. What is the fundamental flaw with the classic "lather-rinse-repeat" hair washing algorithm?

According to the website *wikihow.com*, the lather-rinse-repeat algorithm has two particular flaws: "it doesn't have a condition to end on, and it doesn't really tell you what to repeat. Repeat lathering? Or just the rinsing. A better example would be 'Step 1 - Lather. Step 2 - Rinse. Step 3 - Repeat steps 1 and 2(2 or 3 times for better results) and then finish(exit).' This is understandable by you, has an end condition (a finite number of steps), and is very explicit."

10. Write the pseudocode for an algorithm for washing your hair, that fixes the flaw you describe above.

The following pseudocode represents a loop with a nested loop. This should be printed on shampoo bottles everywhere:

[1] Wet hair thoroughly
[2] *Begin containing wash loop*
[3] *Begin nested lather loop*
[4] Lather with shampoo, repeat lathering as much as you want
[5] *End lather loop*
[6] Rinse with water
[7] Repeat steps #2 through #6 if for some bizarre reason your hair is still dirty
[8] *End wash loop*

11. What happens to a loop when it encounters a break statement?
Loops exit when they encounter a *break* statement. According to our textbook "the *break* statement ends the nearest enclosing *switch* or loop statement" 148.

12. What is the problem with the loop below (* look at the code carefully), and what is the fix (assume *keyboard* has already been properly declared)?

```
boolean keepGoing = true;
while (keepGoing = true)
{
        System.out.print("Enter 0 to quit: ");
        if (keyboard.nextInt() == 0)
        keepGoing = false;
}
```

The problem with the loop is that the conditional statement *while(keepGoing = true)* is actually a variable assignment and not a comparison operator. The correct code would be the following:

```
while(keepGoing == true)
```

13. When are curly braces, { }, required with a loop? When are they not required?
Curly braces are required if the section of code contained within the loop is more than one line. If the section of code is only a single line then curly braces are not required. According to our textbook "a compound statement consisting of a list of statements [must be] enclosed in braces" (139).

14. Write a *heading* for a method named *greet* that returns void and takes no parameters.
The following line of code does this for a public method named *greet*:

```
public void greet()
```

15. Write a *heading* and *body* for a method named *print* that returns void and takes a single String parameter -- label the parameter *message*. In the body of the method, write the Java code to display *message* to the console.
The following line of code produces a heading and body for a public method named *print*:

```
public void print( String message ) {
        System.out.println( message );
}
```