Set 11: *while* Loops

Skill 11.1: Explain the basic structure of a *while* loop
Skill 11.2: Explain the basic structure of a *do-while* loop
Skill 11.3: Explain the advantages of a *do-while* loop

Skill 11.1: Explain the basic structure of a *while* loop

Skill 11.1 Concepts

The *while* loop is basically the same as the *for-loop* you previously learned about. The only difference is that the initializing and step expressions are not part of the while-loop basic structure. The following code illustrates the basic structure of the while loop.

```
while (n <= 79) {
     // some code we want to repeat
}
```

In the above code, notice that the only part similar to the for loop is the control expression, n <= 79. The initializing and step expression are absent. As with the for-loop, the while-loop keeps repeating as long as the control statement is true.

In a previous lesson we used a *for-loop* to sum all the numbers from 3 to 79,

Code

```
int sum = 0;
for (int n = 3; n <= 79; n++) {
     sum += n;
}
System.out.println(sum);
```

Output

```
3157
```

The *for-loop* above requires the following elements to function,

- start = 3
- incrementing = +1
- stop = 79

while loops also require these elements. In fact the *for-loop* above can also be written as a *while* loop as follows,

Code

`int sum = 0;`
`int n = 3;`
 	
`while(n <= 79){`
 	
`  sum += n;`
`  n = +1;`
 	
`}`
 	
`System.out.println(sum);`

Output
3157

Skill 11.1 Exercises 1 and 2

Skill 11.2: Explain the basic structure of a *do-while* loop

Skill 11.2 Concepts

A *do-while* loop is exactly the same as a *while* loop except the control expression is at the bottom of the loop rather than at the top as it is with the *while* loop. The following illustrates the skeleton of a *do-while* loop,

```
do{
//some code we want to repeat
}while(n <= 79);
```

The main difference between the *while* loop and the *do-while* loop is where the test for staying in the loop is made (the control expression)

- *while* loop -> the test is at the top of the loop
- *do-while* loop -> the test is at the bottom of the loop

And, just as we saw that *for-loops* can be written as *while* loops, *for-loops* can also be written as *do-while* loops,

Code

```
int sum = 0;
int n = 3


do{
sum += n;
n = +1;
}while(n <= 79);


System.out.println(sum);
```

Output

3157

Skill 11.2 Exercise 1

Skill 11.3: Explain the advantages of a *do-while* loop

Skill 10.3 Concepts

Depending on the application, do-while loops offer advantages over while loops. Consider for example, an application that prompts a user for input,

Code

```
Scanner s = new Scanner(System.in);
String r = "";
do{
     System.out.println("Do you want to play a game (y)?");
     String r = s.next();
}while(r.equals("y"))
```

Output

Do you want to play a game (y) always displays until the user types something other than y

In the example above, the user is continuously prompted, until a valid input is entered. Although a *while* loop could have also been used, it would have required additional code.

Code

```
Scanner s = new Scanner(System.in);
System.out.println("Do you want to play a game (y)?");
String r = s.next();

while(r.equals("y")){
     System.out.println("Do you want to play a game (y)?");
     r = s.next();
}
```

Output

Do you want to play a game (y) always displays until the user types something other than y

The above example illustrates the utility of a *do-while* loop to check user input. And, conditions that require an action to occur, before the control expression is evaluated.

Skill 11.3 Exercise 1