Repetition using Java – Comparing Loop Structures

With three loop structures available in Java, we need to make decisions about which to use in a given situation. In many cases, this is a matter of personal taste – any loop structure can be used to solve many problems requiring a loop. In most cases, however, a particular looping structure is a more *appropriate* choice.

The `for` statement is the most obvious choice when we know, before starting the loop, how many repetitions we want to occur. The number of repetitions may be fixed (i.e., a number built into the program), or it may be provided by the user, but the value is known before starting the loop.

The `while` and `do` statements are used when we do not know the number of repetitions. Between these two loop structures, recall that the `do` loop will always execute *at least once*, but the `while` loop, depending upon the boolean condition, may not execute at all. Although this is a subtle difference, it may help make the decision between these two options.

Example 1 – All three of the following program fragments have the same effect. They find the sum of the series

$$1+\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}$$

(a) **While Loop** (okay)

```
double sum = 0;
int i = 1;
while (i <= 5)
{
      sum = sum + 1.0/i;    // use 1.0 to make it a double calculation
      i++;
}
```

(b) **Do Loop** (worst because we need to perform `i++` right away)

```
double sum = 0;
int i = 0;
do
{
      i++;                  // have to start with i = 1
      sum = sum + 1.0/i;    // use 1.0 to make it a double calculation
} while (i <= 5);
```

(c) **For Loop** (best because it is very clear that we are looping 5 times)

```
double sum = 0;
int i;
for (i = 1; i <= 5; i++)
{
      sum = sum + 1.0/i;  // use 1.0 to make it a double calculation
}
```

Example 2 – As part of a program to generate report cards, we need a fragment that will read and total the marks. Most students take MAX_SUBJECTS subjects, so we enter marks until we reach this limit. In some cases, students may take less than the maximum, so we can also use the *sentinel value* of -1 to stop mark entry.

In this problem, we loop for MAX_SUBJECTS, which suggests a counted (for) loop. On the other hand, we stop entering marks when the value -1 is entered, which suggests a conditional (while/do) loop. One possible solution to this problem is shown.

```
final int MAX_SUBJECTS = 30;  // a constant, never changes
int totalMarks = 0;       // sum of all marks so far
int numberOfMarks = 0;    // count of mark entries made so far
int mark;
do
{
      System.out.println("Next mark please");
      mark = In.getInt();
      if (mark != -1)
      {
            totalMarks += mark;
            numberOfMarks++;
      }
}
while (mark != -1 && numberOfMarks < MAX_SUBJECTS);
```

Repetition using Java – Comparing Loop Structures

Exercises

1. Describe what the fragment does and the rewrite it using a `for` statement

a)
```
count = 1;
while (count < 10)
{
      System.out.println(count + "   " + Math.sqrt(count));
      count++;
}
```

b)
```
i = 5;
do
{
      i--;
      System.out.println(Math.pow(i,3));
}
while (i > 0);
```

2. Describe what the fragment does and then rewrite it using a `while` statement.

a)
```
for (i = 0; i < 10; i++)
{
      x = 2 * i - 10      // remember order of operations!
      y = Math.abs(x / 5 - 2);
      System.out.println(x + " " + y);
}
```

b)
```
product = 1;
do
{
      System.out.println("Next value please");
      next = In.getInt();
      if (next != TERMINATOR)
      {
            product *= next;
      }
}
while (next != TERMINATOR)
```

3. Describe what the fragment does and then rewrite it using a `do` statement.

a)
```
System.out.println("Enter a value – zero to stop");
value = Math.abs(In.getInt());   // think carefully here!
biggest = value;
while (value != 0)
{
      if (value > biggest)
      {
            biggest = value;
      }
      System.out.println("Enter a value – zero to stop");
      value = Math.abs(In.getInt());
}
```

b)
```
for (i = 0; i == 0;      )
{
      System.out.println("Conintue? (Y/N)");
      response = In.getChar();
      if (response == 'Y' || response == 'N')
      {
            i = 1;
      }
}
```

4. State, with reasons, which loop structure you think would be most appropriate for solving each problem. You do not have to actually solve the problem (unless you are done your work early, in which case it would be a good exercise).

   a) Find the sum of the cubes of the numbers from 1 to 100

   b) Repeatedly prompt the user for a password, rejecting any submissions until the correct password has been provided.

   c) Read and sum positive integers until a sentinel value of -1 has been read.

   d) Determine the number of times that a positive integer can be divided by two.

   e) Find the amount to which $1 will grow in ten years at an interest rate of 8%.