**Chapter 4 Test Review**

**while loops** – Loops until the condition is met. Condition is checked before running <code>

while (condition)

 {

 <code>

 }

 int x = 1;

 while ((0 < x) && (x < 100))

 System.out.println(x++);

int balance = 10;

while (balance >= 1) {

 if (balance < 9) continue;

 balance = balance - 9;

**do…while loops -** Loops until the condition is met. Condition is checked after running <code>

do

 {

 <code>

 }

Code is executed at least once since the condition isn’t checked until after the code is processed.

while and do..while loops are equivalent in their expressive power.

int sum = 0;
int item = 0;
do {
     item++;
     sum += item;
     if (sum >= 4) continue;
}
while (item < 5);

int count = 0;
do {
     System.out.println("Welcome to Java");
} while (count++ < 9);
System.out.println(count);

**for loops** – for loops run for a specified number of iterations

for(condition)

 {

 <code>

 }

for (int i = 5; i > 0; i--) {

 for (int j = 1; j < i; j++)

 System.out.print(j + " ");

 System.out.println();

 }

for (int i = 1; i < 5; i++) {

 for (int j = 1; j < i; j++)

 System.out.print(j + " ");

 System.out.println();

 }

for (int i = 0; i < 15; i++) {

 if (i % 4 == 1)

 System.out.print(i + " ");

 }

1. A variable declared inside a for loop cannot be used outside the for loop.
2. We can use double variables in for..loops, though it does risk some rounding errors if the numbers get too precise.
	1. Example: for(double i = 0.0, i <=1.0, i = i + 0.01)
	2. If you must sum together many number of type double, it is suggest that you add smaller number before bigger numbers as a way to minimize errors.
3. You can always convert a while loop to a for loop.

**Break and Continue**

break has other uses and can be used in situations other than loops.

continue, however is only used inside loops.