Review Questions

Sections 4.2–4.4

4.1 Analyze the following code. Is count < 100 always true, always false, or sometimes true or sometimes false at Point A, Point B, and Point C?

```
int count = 0;
while (count < 100) {
  // Point A
  System.out.println("Welcome to Java!\n");
  count++;
  // Point B
}
  // Point C
```

Sol:

For Point A:

count < 100 always true for Point A, because it is in the while loop and before the increment statement and while loop's execute condition is count < 100. For this reason, it is always true for Point A.

For Point B:

count < 100 is sometimes true sometimes false for Point B. Because it is in the while loop but after the increment statement.

For Point C:

count < 100 always false for Point C, because Point C is come after while loop and while loop's execute condition is count < 100. Therefore, count keep incremented until not been lesser than 100.

4.2 What is wrong if guess is initialized to 0 in line 11 in Listing 4.2?

Sol:

Math.random create a number between 0 and 1. 0 is inclusive, 1 is exclusive. Therefore, if we initialize guess as a 0, while condition might be satisfied already before the first iteration. This cause a logic error, because the program can't reach its purpose.

4.3 How many times is the following loop body repeated? What is the printout of the loop?

```
int i = 1;
while (i < 10)
  if (i % 2 == 0)
    System.out.println(i);
```

(a)

```
int i = 1;
while (i < 10)
  if (i % 2 == 0)
    System.out.println(i++);
```

(b)

```
int i = 1;
while (i < 10)
  if ((i++) % 2 == 0)
    System.out.println(i);
```

(c)

Sol:

(a):

This loop is infinite. i is 1 and 1 always be lesser than 10, but 1 divide by 2 equals to never zero. Therefore, loop continue for infinite time.

(b):

This loop is infinite, too. i is 1 and 1 always be lesser than 10, but 1 divide by 2 equals to never zero. Therefore, i will never be incremented and loop continue for infinite time.

(c):

This loop will execute 9 times, because i will be incremented before module operation. Therefore, loop will be printout these;

3

5

7

9

4.4 What are the differences between a while loop and a do-while loop? Convert the following while loop into a do-while loop.

```
int sum = 0;
int number = input.nextInt();
while (number != 0) {
  sum += number;
  number = input.nextInt();
}
```

Sol:

There is a one major difference in between while and do while loops that loop condition statement comes first in while loops but comes last in do while loops. For this reason, do while loops at least one-time will execute no matter what.

```
int sum = 0;
int number = input.nextInt();
do {
  sum += number;
  number = input.nextInt();
} while ( number!= 0 );
```

4.5 Do the following two loops result in the same value in sum?

```
for (int i = 0; i < 10; ++i) {
  sum += i;
}
```

(a)

```
for (int i = 0; i < 10; i++) {
  sum += i;
}
```

(b)

Sol:

Yes, because third part of for loop condition, evaluate at the end of the loop and value's pre-increment or post-increment doesn't change the inside of the loop.

4.6 What are the three parts of a for loop control? Write a for loop that prints the numbers from 1 to 100.

Sol:

For loop contains three parts. The first part is for initial action, the second part for the loop continuation control - it is decide whether loop continue or not -, the last part execute after each iteration. First and last part generally use for control variable initialize and modify.

```
for (int i = 1 ; i < 101 ; i++ )
{
  System.out.println(i);
}
```

4.7 Suppose the input is 23450. What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    int number, max;
    number = input.nextInt();
    max = number;

    while (number != 0) {
      number = input.nextInt();
      if (number > max)
      max = number;
    }
    System.out.println("max is " + max);
    System.out.println("number " + number);
  }
}
```

Sol:

Program gets input in a loop, compare inputs and determine max input until input is 0. Then prints max and last number. The output will be;

"*max is 5*
number 0"

4.8 Suppose the input is 23450. What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    int number, sum = 0, count;

    for (count = 0; count < 5; count++) {
      number = input.nextInt();
      sum += number;
    }

    System.out.println("sum is " + sum);
    System.out.println("count is " + count);
  }
}
```

Sol:

Program gets input in a loop, summing up the inputs until input is 0. Then prints sum of the inputs and count of the input. The output will be;

"*sum is 14*
count is 5"

4.9 Suppose the input is 2 3 4 5 0. What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    int number, max;
    number = input.nextInt();

    max = number;
```

```
    do {
      number = input.nextInt();
      if (number > max)
      max = number;
    } while (number != 0);

    System.out.println("max is " + max);
    System.out.println("number " + number);
  }
}
```

Sol:

Program gets input in a loop, compare inputs and determine max input until input is 0. Then prints max and last number. The output will be;

“*max is 5*
number 0”

4.10 What does the following statement do?

```
for ( ; ; ) {
do something;
}
```

Sol:

This statement is infinite loop, it does “something” infinite times. Because it hasn’t any control condition.

4.11 If a variable is declared in the for loop control, can it be used after the loop exits?

Sol:

If a variable declared in a loop, it can not use outside the loop.

4.12 Can you convert a for loop to a while loop? List the advantages of using for loops.

Sol:

```
for (int i = 0 ; i < 10 ; i++ )
{
  do something;
}
////////////////////////////

int i = 0;
while ( i < 10 )
{
  do something;
  i++;
}
```

For loops, more compact than while loops. And using for loops is a better way to avoid some simple mistakes like forget incrementing or decrementing control value after the loops. It's a better way to use it if you know how many repetitions to do it.

4.13 Convert the following for loop statement to a while loop and to a do-while loop:

```
long sum = 0;
for (int i = 0; i <= 1000; i++)
  sum = sum + i;
```

Sol:

```
// while loop
long sum = 0;
int i = 0;
while ( i <= 1000 )
{
  sum += i;
  i++;
}
```

```
// do-while loop
long sum = 0;
int i = 0;
do{
  sum += i;
  i++;
} while ( i <= 1000 );
```

4.14 Will the program work if n1 and n2 are replaced by n1 / 2 and n2 / 2 in line 17 in Listing 4.8?

Sol:

No, when n1 = 3 and n2 = 3 program will not work properly.

4.15 What is the keyword break for? What is the keyword continue for? Will the following program terminate? If so, give the output.

```
int balance = 1000;
while (true) {
  if (balance < 9)
    break;
  balance = balance - 9;
}

System.out.println("Balance is "
  + balance);
```

(a)

```
int balance = 1000;
while (true) {
  if (balance < 9)
    continue;
  balance = balance - 9;
}

System.out.println("Balance is "
  + balance);
```

(b)

Sol:

We are using break keyword for terminating the loop, and continue keyword for terminating the spesific iteration and continue next one.

a's output will be `Balance is 1`.

But b will not have any output because continue keyword cause infinite loop in b. Therefore, compiler will give unreachable statement error.

4.16 Can you always convert a while loop into a for loop? Convert the following while loop into a for loop.

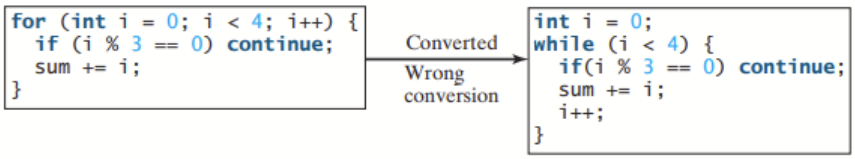
```
int i = 1;
int sum = 0;
while (sum < 10000) {
  sum = sum + i;
  i++;
}
```

Sol:

We can always convert while loop into a for loop. Because for loop's features covers while loop.

```
int sum = 0;
for ( int i = 1 ; sum < 10000 ; i++ )
  sum += i;
```

4.17 The for loop on the left is converted into the while loop on the right. What is wrong? Correct it.



Sol:

In while loop when i == 3, loop always continue from next iteration, but in next iteration i will be 3 again because program never reach `i++;`, therefore this cause infinite loop.

This is the correct version;

```
int i = 0;
while (i < 4)
{
 if( i % 3 == 0 )
  {
    i++;
    continue;
  }
 sum += i;
 i++;
}
```

4.18 Rewrite the programs TestBreak and TestContinue in Listings 4.11 and 4.12 without using break and continue.

Sol:

```
public class TestBreak
{
  public static void main ( String[] args )
  {
    int sum = 0;
    int number = 0;
    int temp = 0;

    while ( number < 20 )
    {
      number++;
      temp = number;
      sum += number;
      if ( sum >= 100 )
        number = 20;
    }

    number = temp;

    System.out.printf( "The number is %d\n", number );
    System.out.printf( "The sum is %d", sum );
  }
}
```

```
public class TestContinue
{
```

```
   public static void main ( String[] args )
   {
     int sum = 0;
     int number = 0;

     while ( number < 20 )
     {
       number++;
       if ( number == 10 || number == 11 )
         number = 12;
       sum += number;
     }

     System.out.printf( "The sum is %d" , sum);
   }
 }
```

4.19 After the break statement is executed in the following loop, which statement is executed? Show the output.

```
for (int i = 1; i < 4; i++) {
  for (int j = 1; j < 4; j++) {
    if (i * j > 2)
      break;
    System.out.println(i * j);
  }
  System.out.println(i);
}
```

Sol:

After the break statement, `System.out.println(i);` will be executed and output will be;

1
2
1
2
2
3

4.20 After the continue statement is executed in the following loop, which statement is executed? Show the output.

```
for (int i = 1; i < 4; i++) {
  for (int j = 1; j < 4; j++) {
    if (i * j > 2)
      continue;
    System.out.println(i * j);
  }
  System.out.println(i);
}
```

Sol:

After the break statement, `j++` statement will be executed and output will be;

1
2
1
2
2
3

Comprehensive

4.21 Identify and fix the errors in the following code:

```
public class Test {
  public void main(String[] args) { // missing static
    for (int i = 0; i < 10; i++); // the semicolon should be removed
      sum += i; // sum not defined

    if (i < j); // the semicolon should be removed and j not defined
      System.out.println(i) // missing semicolon
    else
      System.out.println(j);

    while (j < 10); // the semicolon should be removed
    {
      j++;
    };

    do {
      j++;
    } while (j < 10) // missing semicolon
  }
}
```

Sol:

```
public class Test
{
  public static void main( String[] args )
  {
    int i = 0;
    int j = 0;
    int sum = 0;

    for ( ; i < 10 ; i++)
      sum += i;

    if ( i < j )
      System.out.println( i );
    else
      System.out.println( j );

    while ( j < 10 )
    {
      j++;
    }

    do {
      j++;
    } while ( j < 10 );
  }
}
```

4.22 What is wrong with the following programs?

```
1 public class ShowErrors {
2   public static void main(String[] args) {
3     int i;
4     int j = 5;
5
6     if (j > 3)
7       System.out.println(i + 4);
8   }
9 }
```

(a)

```
1 public class ShowErrors {
2   public static void main(String[] args) {
3     for (int i = 0; i < 10; i++);
4       System.out.println(i + 4);
5   }
6 }
```

(b)

Sol:

(a):

i did not initialize. Therefore, compiler gives syntax error.

(b):

Using ";" semicolon at the end of the for statement cause the empty loop. For this reason, i will have been initialized in the loop and it can't accessible outside the loop. So, `System.out.println()` statement can't find i and compiler gives syntax error.

4.23 Show the output of the following programs. (Tip: Draw a table and list the variables in the columns to trace these programs.)

```
public class Test {
  /** Main method */
  public static void main(String[] args) {
    for (int i = 1; i < 5; i++) {
      int j = 0;
      while (j < i) {
        System.out.print(j + " ");
        j++;
      }
    }
  }
}
```

(a)

```
public class Test {
  /** Main method */
  public static void main(String[] args) {
    int i = 0;
    while (i < 5) {
      for (int j = i; j > 1; j-)
        System.out.print(j + " ");
      System.out.println("****");
      i++;
    }
  }
}
```

(b)

```
public class Test {
  public static void main(String[] args) {
    int i = 5;
    while (i >= 1) {
      int num = 1;
      for (int j = 1; j <= i; j++) {
        System.out.print(num + "xxx");
        num *= 2;
      }

      System.out.println();
      i--;
    }
  }
}
```

(c)

```
public class Test {
  public static void main(String[] args) {
    int i = 1;
    do {
      int num = 1;
      for (int j = 1; j <= i; j++) {
        System.out.print(num + "G");
        num += 2;
      }

      System.out.println();
      i++;
    } while (i <= 5);
  }
}
```

(d)

Sol:

(a):

`0 0 1 0 1 2 0 1 2 3`

(b):

There is a typo end of the for statement. It should be `j--` not `j-`. I correct that and the output is:

```
****
****
2 ****
3 2 ****
4 3 2 ****
```

(c):

```
1xxx2xxx4xxx8xxx16xxx
1xxx2xxx4xxx8xxx
1xxx2xxx4xxx
1xxx2xxx
1xxx
```

(d):

```
1G
1G3G
1G3G5G
1G3G5G7G
1G3G5G7G9G
```

4.24 What is the output of the following program? Explain the reason.

```
int x = 80000000;
while (x > 0)
  x++;
System.out.println("x is " + x);
```

Sol:

`x is -2147483648`

This loop keep increments x value until it is not bigger than 0. When x reach (2^32 - 1) (biggest number it can reach as an integer) loop increments x value one last time. This cause overflow and x will be -2147483648. This value can't satisfy while's condition, therefore, x's value will be printed of the last line.

4.25 Count the number of iterations in the following loops.

```
int count = 0;
while (count < n) {
  count++;
}
```

(a)

```
for (int count = 0;
   count <= n; count++) {
}
```

(b)

```
int count = 5;
while (count < n) {
  count++;
}
```

(c)

```
int count = 5;
while (count < n) {
  count = count + 3;
}
```

(d)

Sol:

(a): n times.

(b): n+1 times.

(c): n-5 times.

(d): ceiling rounded of the ((n-5)/3) times