Review Questions

Sections 5.2–5.4

5.1 What are the benefits of using a method? How do you define a method? How do you invoke a method?

- Methods make our programs more modular, reusable, readable and understandable.
- We define a method with creating method signature and method body. This means;

```
modifier(s) returnValueType methodName (list of parameters)
{
  // Method Body
}
```

- We can invoke a method with two ways. First use as an expression, seconds use as a statement;

```
public static void main( String[] args)
{
  int a = 3;
  int b = 4;

  int c = Max( a , b ); // Expression use

  Max( a , b ); // Statement use
}
public int Max ( int n1 , int n2 )
{
  int result;
  if ( n1 > n2 )
    result = n1;
  else
    result = n2;

  return result;
}
```

5.2 What is the return type of a main method?

- Main method don't return anything, so `void`.

5.3 Can you simplify the max method in Listing 5.1 using the conditional operator?

```
/** Return the max between two numbers */
public static int max( int num1 , int num2 )
{
  return ( num1 > num2 ) ? num1 : num2;
}
```

5.4 True or false? A call to a method with a void return type is always a statement itself, but a call to a value-returning method is always a component of an expression

- True, we can call a void return type method only as a statement
- False, we can call value-returning method as a statement. In this situation, we just don't use return value.

5.5 What would be wrong with not writing a return statement in a value-returning method? Can you have a return statement in a void method? Does the return statement in the following method cause syntax errors?

```
public static void xMethod(double x, double y) {
  System.out.println(x + y);
  return x + y;
}
```

- We will have been receiving a compile/syntax error because of not using the return statement.
- You can use return statement itself in a void method as a just itself (`return;`) . This can be useful, especially, if we will want to terminate the method in a particular situation.
- We will have been receiving a compile/syntax error because of trying to return a value, writing a value after the return statement in a void method.

5.6 Define the terms parameter, argument, and method signature.

- The variables defined in the method header are known as **parameters**. When a method is invoked, we pass a value to the parameter. This passing value (already created variable or a literal) is calling as an **argument**.
- **Method signature** is method's definition statement. It contains method's modifiers, return value type, method name and parameters list.

5.7 Write method headers for the following methods:

▼ Computing a sales commission, given the sales amount and the commission rate.

```
public static double ComputeSalesCommision( double salesAmount , double commisionRate )
```

▼ Printing the calendar for a month, given the month and year.

```
public static void PrintCalendarOfAMonth( int month , int year )
```

▼ Computing a square root.

```
public static double ComputeSquareRoot( double number )
```

▼ Testing whether a number is even, and returning true if it is.

```
public static boolean IsEven( int number )
```

▼ Printing a message a specified number of times.

```
public static void PrintMessageDesiredTimes( String message , int times )
```

▼ Computing the monthly payment, given the loan amount, number of years, and annual interest rate.

```
public static double ComputeMonthlyPayment( double loanAmount , int numberOfYears , double interestRate )
```

▼ Finding the corresponding uppercase letter, given a lowercase letter

```
public static char MakeUppercase( char letter )
```

5.8 Identify and correct the errors in the following program

```
1 public class Test {
2   public static method1(int n, m) {
3     n += m;
4     method2(3.4);
5   }
6
7   public static int method2(int n) {
8     if (n > 0) return 1;
9     else if (n == 0) return 0;
10    else if (n < 0) return -1;
11  }
12 }
```

```
public class Test
{
  public static void method1 ( int n , int m )
  {
    n += m;
    method2( 3.4 );
  }

  public static int method2 ( double n )
  {
    if( n > 0 )
      return 1;
    else if ( n == 0 )
      return 0;
    else
      return -1;
  }
}
```

5.9 Reformat the following program according to the programming style and documentation guidelines proposed in §2.16, “Programming Style and Documentation.” Use the next-line brace style.

```
public class Test {
  public static double method1(double i,double j)
  {
  while (i<j) {
    j--;
  }

  return j;
  }
}
```

```
public class Test
{
  public static double method1( double i , double j)
  {
    while( i < j )
      j--;

    return j;
  }
}
```

Sections 5.5–5.7

5.10 How is an argument passed to a method? Can the argument have the same name as its parameter

- To pass an argument to a method, we must pass a compatible type of value in the right order to method.
- Yes, because parameter exist in only the local scope of method. And parameter take argument's value, not argument itself into the method.

5.11 What is pass-by-value? Show the result of the following programs:

```
public class Test {
  public static void main(String[] args) {
    int max = 0;
    max(1, 2, max);
    System.out.println(max);
  }

  public static void max(
      int value1, int value2, int max) {
    if (value1 > value2)
      max = value1;
    else
      max = value2;
  }
}
```

(a)

```
public class Test {
  public static void main(String[] args) {
    int i = 1;
    while (i <= 6) {
      method1(i, 2);
      i++;
    }
  }

  public static void method1(
      int i, int num) {
    for (int j = 1; j <= i; j++) {
      System.out.print(num + " ");
      num *= 2;
    }

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

(b)

```
public class Test {
  public static void main(String[] args) {
    // Initialize times
    int times = 3;
    System.out.println("Before the call,"
      + " variable times is " + times);

    // Invoke nPrintln and display times
    nPrintln("Welcome to Java!", times);
    System.out.println("After the call,"
      + " variable times is " + times);
  }

  // Print the message n times
  public static void nPrintln(
      String message, int n) {
    while (n > 0) {
      System.out.println("n = " + n);
      System.out.println(message);
      n--;
    }
  }
}
```

(c)

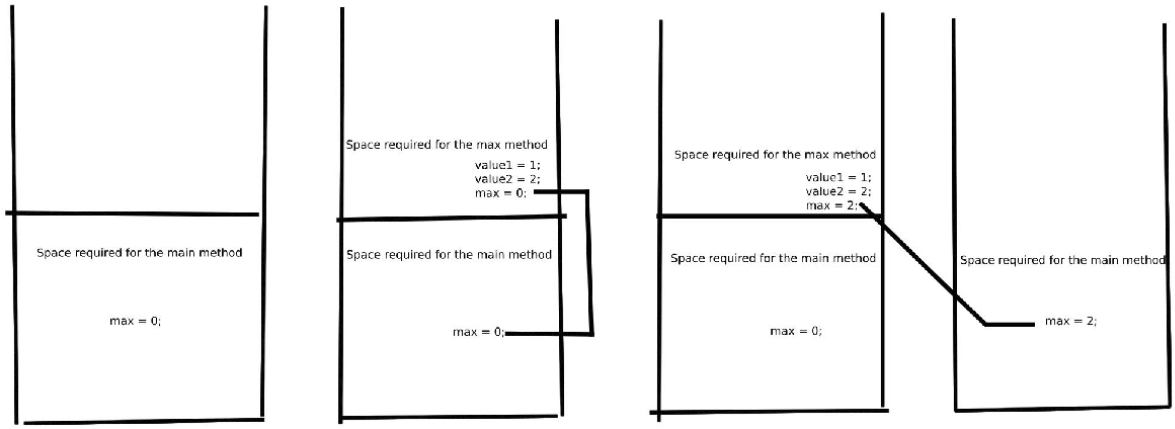
```
public class Test {
  public static void main(String[] args) {
    int i = 0;
    while (i <= 4) {
      method1(i);
      i++;
    }

    System.out.println("i is " + i);
  }

  public static void method1(int i) {
    do {
      if (i % 3 != 0)
        System.out.print(i + " ");
      i--;
    }
    while (i >= 1);

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

(d)

- When we invoke a method and give to it an argument, method doesn't take argument itself, takes argument's value and assign to parameter. This is mean *pass-by-value.*

- (a) : 0
- (b):
  2
  2 4
  2 4 8
  2 4 8 16
  2 4 8 16 32
  2 4 8 16 32 64
- (c):
  Before the call, variable times is 3
  n = 3
  Welcome to Java!
  n = 2
  Welcome to Java!
  n = 1
  Welcome to Java!
  After the call, variable times is 3
- (d):
  1
  2 1
  2 1
  4 2 1
  i is 5

5.12 For (a) in the preceding question, show the contents of the stack just before the method max is invoked, just as max is entered, just before max is returned, and right after max is returned



Section 5.8

5.13 What is method overloading? Is it permissible to define two methods that have the same name but different parameter types? Is it permissible to define two methods in a class that have identical method names and parameter lists but different return value types or different modifiers?

- Method overloading, creating another method with same method name and modifiers but different parameter list.

- You can’t overload a method that differences based on return type or modifiers.

5.14 What is wrong in the following program?

```
public class Test {
  public static void method(int x) {
  }

  public static int method(int y) {
    return y;
  }
}
```

- We can’t overload a method based on its return type. If we want to change return type, we must change parameters too.

Section 5.9

5.15 Identify and correct the errors in the following program:

```
1 public class Test {
2   public static void main(String[] args) {
3     nPrintln("Welcome to Java!", 5);
4   }
5
6   public static void nPrintln(String message, int n) {
7     int n = 1;
```



```
 8     for (int i = 0; i < n; i++)
 9       System.out.println(message);
10   }
11 }
```

```
public class Test
{
  public static void main( String[] args )
  {
    nPrintln( "Welcome to Java!" , 5 );
  }

  public static void nPrintln( String message, int n )
  {
    /* n is a parameter and don't needed to be declaring again.
      And assigning 1 to it, just ignores the argument.
      We have to remove both of them.
    */
    for( int i = 0 ; i < n ; i++)
      System.out.println( message );
```

```
  }
}
```

Section 5.10

5.16 True or false? The argument for trigonometric methods represents an angle in radians.

- True

5.17 Write an expression that returns a random integer between 34 and 55. Write an expression that returns a random integer between 0 and 999. Write an expression that returns a random number between 5.5 and 55.5. Write an expression that returns a random lowercase letter.

- `34 + ( int )( Math.random() * 21 );`
- `( int )( Math.random() * 999 );`
- `5.5 + ( Math.random() * 50 );`
- `( char )('a' + Math.random() * ( 'z' - 'a' + 1 ) );`

5.18 Evaluate the following method calls:

(a) Math.sqrt(4) → 2.0
(b) Math.sin(2 * Math.PI) → 0 (It will be 0 only if we cast to int).
(c) Math.cos(2 * Math.PI) → 1.0
(d) Math.pow(2, 2) → 4.0
(e) Math.log(Math.E) → 1.0
(f) Math.exp(1) → 2.718281828459045
(g) Math.max(2, Math.min(3, 4)) → 3
(h) Math.rint(-2.5) → -2.0
(i) Math.ceil(-2.5) → -2.0
(j) Math.floor(-2.5) → -3.0
(k) Math.round(-2.5F) → -2
(l) Math.round(-2.5) → -2
(m) Math.rint(2.5) → 2.0
(n) Math.ceil(2.5) → 3.0
(o) Math.floor(2.5) → 2.0
(p) Math.round(2.5F) → 3
(q) Math.round(2.5) → 3
(r) Math.round(Math.abs(-2.5)) → 3