Quiz 2

• 20 points possible
• Average: 14.7
• Min: 9
• Max: 20
• Median: 15

Question 1 or 2

1. (2 points) What are the values of a and b after the following program fragment executes?

```java
int a = 1;
int b = 0;
while (a < 6) {
    b = b + a;
    a++;
}
a = 6 b = 15
```
2. (4 points) If a = false, b = true, c = 10 and d = 20, what are the values of these expressions?

- !a || b = true
- !(a || b) = false
- (c < d) && b = true
- (c == 10) = true

3. (3 points) Rewrite the following for loop to use a while statement:

```java
for (int i = 0; i < 100; i++) { // or 50
    sum = sum + i;
}
```

Answer:

```java
int i = 0;
while (i < 100) { // or 50
    sum = sum + i;
    i++;
}
```

4. (2 points) What is printed by the following program fragment? Be careful – there's a common programming error here:

```java
int x = 5; // or int x = 3;
int y = 6; // or int x = 4;
if (x < y);
    System.out.println(x + " is smaller");
if (y < x);
    System.out.println(y + " is smaller");
```

Answer:

5 is smaller // 3 is smaller
6 is smaller // 4 is smaller
Question 5

5. (4 points) Write a program fragment that checks to see if an integer variable \( a \) has a value less than 5. If \( a \) is less than 5, print "Yes". Otherwise, print "No".

```java
int a = 0;
if ( a < 5 ) // or 10
    System.out.println("Yes");
else
    System.out.println("No");
```

Question 6

6. (2 points) What is printed by the following program fragment? Be careful.

```java
int a = 3; // or int a = 5;
int b = 2;
if ( a == 1 )
    if ( b == 2 )
        System.out.println("b is 2");
else
    System.out.println("a is not 1");
System.out.println("a = " + a );
```

Answer:

```
a = 3
```

```
a = 5
```

Question 7

7. (3 points) What does the following program fragment print? Recall that \% is the modulus or remainder operator.

```java
for (int i = 1; i <= 20; i++) // or 25
    if (i % 4 == 0) // or 5
        System.out.print(i + " ");
System.out.println();
```

Answer:

```
4 8 12 16 20 5 10 15 20 25
```
Last Time

// Message.java: Simple method use
class Message {
    public static void main(String[] args) {
        System.out.println("Hello, class!");
        printMessage(); // method call
        System.out.println("GoodBye.");
    }
    // definition of method printMessage
    static void printMessage() {
        System.out.println("A message for you:");
        System.out.println("Have a nice day!\n");
    }
}

Defining Simple Methods

public static ReturnType Identifier (ParameterList) {
    Body
}

- **ReturnType** is the type of value returned from the method/function.
- **Identifier** is the name of the method/function.
- **ParameterList** is a list of variable declarations that will be used to pass information into the method. These are called the *formal parameters*.
- **Body** is a list of statements and declarations describing the action performed by this method.
- For the time being
  - **public** is optional
  - **static** is required

Void return type

- Some methods don't return anything
  - For example, System.out.println()
- Use the keyword **void** to declare these methods
  - static void PrintMessage()
Parameters and Arguments

- When you define a method, you specify the *formal parameters* that the method expects to receive.
  - The formal parameters are like variable declarations within the method:
    - They have a type
    - They have a name
    - They can be used like any other variable
- A method may have no parameters:
  - static void PrintMessage()

Parameters and Arguments

- When you call the method, you pass it *actual parameters or arguments*:
  - These are the values you would like the method to use when you call it.
- When you call a method, you must have the same number and type of parameters as in the definition.
- Arguments are passed to the method by their position, not their identifier.

Parameters and Arguments: Examples

- MethodExample.java – a method with two arguments
- MethodExample2.java – call the method using variables instead of literals
- MethodExample3.java – incorrect number of parameters
Parameters and Arguments: Examples

- MethodExample4.java - The variables used for actual and formal parameters have the same names
- MethodExample5.java – Don’t confuse the names of the actual and formal parameters
- MethodExample6.java – Don’t confuse the names of the actual and formal parameters

Don’t forget static

- Until chapter 6, all of the methods you write should have the qualifier static. If you leave it off you will get a message like:

Can’t make static reference to method returnType methodName (...) in class YourClass.

- See MethodExample7.java
- I have added a link on the homework page on the course web site to “TA On-line” – good explanations for Java compiler messages

Return

- So far, all of our examples have used void as the return type
  - Method automatically returns to the caller when it reaches the end
  - No value is returned since the method type is void
Return

- Methods that have a non-void return type return a value to the caller
  - `x = Math.sqrt()`
- The `return` statement is used to return values to the caller
  ```java
  return 3;
  return true;
  return x;
  return ( (a + b)/2.0 );
  return;
  ```
- The type of the expression in the `return` statement must be the same as the method return type

// Min2.java: return expression in a method

class Min2 {
    public static void main(String[] args) {
        int j = 78, k = 3 * 30, m;
        System.out.println("Minimum of two integers Test:");
        m = min(j, k);
        System.out.print("The minimum of : ");
        System.out.print( j +", "+ k +" is "+ m);
    }
    static int min(int a, int b) {
        if (a < b)
            return a;
        else if (b < a)
            return b;
        System.out.println("they are equal!!!");
        return a;
    }
}

More about return

Control returns immediately from a method when a return is executed.