Chad Philip Johnson
CSCI20, Worthington
October 09th, 2012

Pencil & Paper 08

1. Given the following array declaration and method, write the code to invoke the method, passing in the first element of the array to the method:

```
        Plant myPlants[] = new Plant[10];
        public void waterPlant (Plant aPlant)
        {
             aPlant.addWater(10);
        }
```

The following line of code performs this operation:

```
        waterPlant( myPlants[0] );
```

2. Write a for loop that will pass each element of the array myPlants above into the method waterPlant defined above.

The following lines of code will perform this operation:

```
        for( i = 0; i < myPlants.length; i++ ) {
                waterPlant( myPlants[i] );
        }
```

3. Write a function definition for a function named doubleAll that takes an integer array numbers as its parameter. The function should loop through the array, multiple each element by 2, and store the new value back into the array.

The following lines of code represent this function:

```
        public static void doubleAll( int[] intArray )      {
                for( int i = 0; i < intArray.length; i++ )    {
                        intArray[i] *= 2;
                }
        }
```

4. Assume that you have a Java bytecode file named Process.class that contains a complete Java program. Launch the program using the Java Virtual Machine (JVM) and pass the arguments "hello everyone how are you" (without the quotes) into the program as it is launched.

The following command would launch the program and pass the intended strings as arguments:

```
        java Process hello everyone how are you
```

5. Fill in the main method below with the code to display each element of the args array to the screen.

```
public static void main (String [ ] args)
{       }
```

The following code will perform these tasks:

```
        if( args.length > 0 )     {
                for( String word : args ) {
                        System.out.printf( "%s\n", word );
                }
        }
```

6. What is the output of the code you wrote in #5 above, if this code were in the program Process referenced above, and you launched the program with the arguments specified in #4 above?

The output would be the following:

```
hello
everyone
how
are
you
```

7. Explain the relationship between the two arrays referenced in the code below.

```
int [ ] firstList = {10, 20, 30, 40, 50};
int [ ] secondList = firstList;
```

Both variables firstList and secondList contain the same address to where the array object (defined by {10, 20, 30, 40, 50}) exists in memory. The assignment in the second line of code above simply gives the array variable secondList the same memory address as the array variable firstList. Only one array object actually exists.

8. What is the output of the following code (firstList and secondList refer to the arrays declared above)?

```
for (int i=0; i<firstList.length; i++)
     firstList[i] = firstList[i] * 10;
for (int i=0; i<secondList.length; i++)
     System.out.print(secondList[i] + ",");
```

The output is the following:

```
100,200,300,400,500,
```

9. What is the output of the following code (firstList and secondList refer to the arrays declared above)?

```
if (firstList == secondList)
     System.out.println("They are the same array.");
else
     System.out.println("They are NOT the same array.");
```

The output is the following:

```
They are the same array.
```

10. Write the code for a method named getEvens that accepts an integer array named numberList as its parameter and returns an integer array. The method should do the following:

- declare and initialize to zero an int variable named evenCount

- use a for loop to loop through the passed-in array and increment evenCount for every even number found

- declare a new array named evenNumbers that is sized to the value of evenCount

- declare and initialize to zero an int variable named nextEvenNumber

- loop through the array numberList and store each even number found into array evenNumbers; use nextEvenNumber as the index value for evenNumbers, and increment nextEvenNumber each time an even value is stored into evenNumbers

- return evenNumbers

The following method will perform the requested operations:

```
public static int[] getEvens( int[] someArray )     {

        int evenCount= 0;
        for( int i = 0; i < someArray.length; i++ )   {

            if( (someArray[i] % 2) == 0 )    {      evenCount++; }

        }

        int[] evenNumbers   = new int[evenCount];
        int nextEvenNumber  = 0;

        for( int i = 0; i < someArray.length; i++ )   {

            if( (someArray[i] % 2) == 0 )    {
                evenNumbers[nextEvenNumber++]    = someArray[i];     }

        }

        return evenNumbers;

    }
```