Chad Philip Johnson
CSCI20, Worthington
October 05th, 2012

Pencil & Paper 07

1. How does your text define array?
According to our textbook, an array "is a data structure used to process a collection of data that is all of the same type, such as a list of numbers of type double or a list of strings" (346). It also mentions that an array "behaves like a list of variables with a uniform naming mechanism that can be declared in a single line of simple code" (346).

2. According to your text, an array object is different from other objects you have encountered in what two ways?
According to our textbook, arrays are viewed "as a collection of indexed variables and as a single item whose value is a collection of values of the base type" (356). Arrays are collections of variables in that a single variable name along with an index value can access many related values of the same type. Like other objects, arrays are reference types and contain the address of where the array object is stored in memory. However, this address represents the location in memory where the array's collection of grouped values are stored.

3. Declare an array of type *double* of size 31. Name the array *januaryTemperatures*.
The following line of code will declare an array of type double size 31:

```
double[] januaryTemperatures = new double[31];
```

4. Assign the value 30.4 to the first day of the array *januaryTemperatures* that you declared above.
The following line of code performs this assignment:

```
januaryTemperatures[0] = 30.4;
```

5. Write a *for* loop to prompt for and read in the temperature values for the remaining days for *januaryTemperatures* (exclude the first day, since that has been assigned above), using an instance of CinReader named reader. Assume that the code 'CinReader reader = new CinReader();' has already been written.
The following lines of code perform this task:

```
for( int i = 1; i < januaryTemperatures.length; i++ ) {
      januaryTemperatures[i] = reader.readDouble();
}
```

6. Fill in the following blanks with the answers appropriate to the following array declaration:

int [ ] movieRatings = new int[100];
array base type: int
array name: movieRatings
array size: 100
first element index: 0
last element index: 99

7. Write a *for* loop to display all of the values in the array *myFavoriteMovies* to the console.

The following lines of code perform this task:

```
for( int i = 0; i < myFavoriteMovies.length; i++ )  {
      System.out.printf( "[%d] %f\n", i, myFavoriteMovies[i] );
}
```

8. Write a *for-each* statement to create a display using all of the values from the int array wins. The display should say "Hooray!" if the wins value is greater than 10, otherwise the display should say "Boo!".

The following lines of code perform this task:

```
for( int score : wins )   {
      if( score > 10 )    {
            System.out.printf( "Hooray!  %d\n", score );
      else
            System.out.printf( "Boo!  %d\n", score );
      }
}
```

9. What is the name for the error that occurs when an array index is used that is less than zero or greater than or equal to the size of the array?

This is called the *array index out of bounds* exception. According to our textbook, this error occurs when "an index expression evaluates to some value other than those allowed by the array declaration" 352.

10. Does the following code cause the error described above? Explain your answer.

```
char [ ] letters = new char[26];
letters[26] = 'Z';
```

Yes, this produces the *array index out of bounds* exception. The first index value for the array *letters* is 0 and the final index value is 25, representing an array of total length 26. The index number 26 represents the 27th char in the array letters, which does not exist.

11. Declare and initialize, in a single line of Java code, a String array named *eightball* with the answers "yes", "no", "maybe", and "never".

The following line of code performs this task:

```
String[] eightball = new String[] { "yes", "no", "maybe", "never" };
```

12. What output is produced by the following code?

```
int [ ] numbers = {3,5,7,9,11};
for (int n : numbers)
{
     if (n % 3 == 0)
           System.out.println(n + " is divisible by three");
}
```

The output is as follows:

3 is divisible by three
9 is divisible by three

13. What output is produced by the following code?

```
boolean [ ] flags = {true, false, true, false, true, false};
for (int i=0; i<flags.length; i+=2)
{
     if (flags[i] == true)
```

```
                System.out.print("yes,");
            else
                System.out.print("no,");
        }
```

The output is as follows:

yes,yes,yes,

14. What is a *local variable*? Can two different methods have local variables with the same name?

A local variable is a variable declared within a method and is visible only within the containing method. It can have the same name as a field in the class or as a field in another method. According to our textbook a local variable is "a variable declared within a method. It is called *local* because is meaning is local to—that is, confined to—the method definition. If you have two methods and each of them declares a variable of the same name—for example, if both were named keyboard—they would be two different variables that just happen to have the same name. Any change that is made to the variable named keyboard within one method would have no effect upon the variable named keyboard in the other method" (186).

15. What is a formal parameter for a method? What is an argument to a method?

According to our textbook, a formal parameter is "like a blank that is filled in with a particular value when the method is invoked" (188). An argument is directly related to the parameter and is "the value that is plugged in for the parameter" (188-189). Methods contain parameters whenever additional values are needed from another part of the program in order to perform its operations. The particular values passed to the method are its arguments, and this occurs whenever the method is called.