Review Questions

Section 6.2

6.1 | How do you declare and create an array?

We can declare an array with using this syntax `elementType[] arrayRefVar;` .

We can create an array with using this syntax `arrayRefVar = new elementType[ arraySize ];` .

6.2 | How do you access elements of an array?

We can access elements of an array with this syntax `array[ indexValue ]` . "*array*" is the name of our array and "*indexValue*" is which number of value we want to access. Index starts 0 and goes until array's length - 1.

6.3 | Is memory allocated for an array when it is declared? When is the memory allocated for an array? What is the printout of the following code?

There are no allocated memory when array declared. Memory allocate if we create an array.

Output is :

x is 60
The size of numbers is 30

6.4 | Indicate true or false for the following statements:

- Every element in an array has the same type. → **True**
- The array size is fixed after it is declared. → **False**
- The array size is fixed after it is created. → **True**
- The elements in an array must be of primitive data type. → **False**

6.5 | Which of the following statements are valid array declarations?

- int i = new int(30); → **Invalid,** must be → **int[] i** or **int i[]**
- double d[] = new double[30]; → **Valid**
- char[] r = new char(1..30); → **Invalid,** must be → **new char[1]** or **new char[30]**
- int i[] = (3, 4, 3, 2); → **Invalid,** must be → { 3 , 4 , 3 , 2 };
- float f[] = {2.3, 4.5, 6.6}; → **Valid,**
- char[] c = new char(); → **Invalid,** must be → **new char[indexValue];**

6.6 | What is the array index type? What is the lowest index? What is the representation of the third element in an array named a?

Array index type is **int** and lowest index type is **0**.

Third element in an array can represent with **a[2]**.

6.7 | Write statements to do the following:

a. Create an array to hold 10 double values.

```
double[] arr = new double[ 10 ];
```

b. Assign value 5.5 to the last element in the array.

```
arr[ arr.length - 1 ] = 5.5;
```

c. Display the sum of the first two elements.

```
double sumOfFirstTwoElements = arr[ 0 ] + arr[ 1 ];
System.out.println( sumOfFirstTwoElements );
```

d. Write a loop that computes the sum of all elements in the array.

```
double sumOfArray = 0;

for( int i = 0 ; i < arr.length ; i++ )
{
  sumOfArray += arr[i];
}

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

e. Write a loop that finds the minimum element in the array.

```
double min = arr[0];
for( int i = 0 ; i < arr.length; i++ )
{
    if( min > arr[i])
      min = arr[i];
}
```

f. Randomly generate an index and display the element of this index in the array.

```
int randomIndex = ( int )( Math.random() * arr.length );

System.out.println( arr[ randomIndex ] );
```

g. Use an array initializer to create another array with initial values 3.5, 5.5, 4.52, and 5.6.

```
double[] anotherArr = { 3.5 , 5.5 , 4.52 , 5.6 };
```

6.8 | What happens when your program attempts to access an array element with an invalid index?

Compiler will be give **ArrayIndexOutOfBounds** error**.**

6.9 | Identify and fix the errors in the following code

```
public class Test
{
  public static void main(String[] args)
  {
    double[100] r; // this initialize is wrong usage of syntax. array was not created.

    for (int i = 0; i < r.length(); i++); // r.length() usage and semicolon(;) place are wrong usage of syntax
      r(i) = Math.random * 100; // r(i) usage and using Math.random without bracets are wrong usage of syntax
  }
}
```

```
public class Test
{
  public static void main(String[] args)
  {
    double[] r = new double[100];

    for (int i = 0; i < r.length; i++)
      r[i] = Math.random() * 100;
  }
}
```

Section 6.3

6.10 | Use the arraycopy() method to copy the following array to a target array t:

```
public class Test
{
  public static void main(String[] args)
  {
    int[] source = {3, 4, 5};
    int[] duplicateArr = new int[3];

    System.arraycopy( source , 0 , duplicateArr , 0 , source.length );

    for (int i = 0; i < duplicateArr.length; i++)
      System.out.println(duplicateArr[i]);
  }
}
```

6.11 | Once an array is created, its size cannot be changed. Does the following code resize the array?

It is creates a new array and assign new array address's to myList.

6.12 | When an array is passed to a method, a new array is created and passed to the method. Is this true?

False. When an array is passed to a method, the method takes the parameter's value, which is array's address, and creates a new array reference variable and assign array's address to this reference variable.

6.13 | Show the output of the following two programs:

```java
public class Test {
  public static void main(String[] args) {
    int number = 0;
    int[] numbers = new int[1];

    m(number, numbers);

    System.out.println("number is " + number
      + " and numbers[0] is " + numbers[0]);
  }

  public static void m(int x, int[] y) {
    x = 3;
    y[0] = 3;
  }
}
```

a

```java
public class Test {
  public static void main(String[] args) {
    int[] list = {1, 2, 3, 4, 5};
    reverse(list);
    for (int i = 0; i < list.length; i++)
      System.out.print(list[i] + " ");
  }

  public static void  reverse(int[] list) {
    int[] newList = new int[list.length];

    for (int i = 0; i < list.length; i++)
      newList[i] = list[list.length - 1 - i];

    list = newList;
  }
}
```

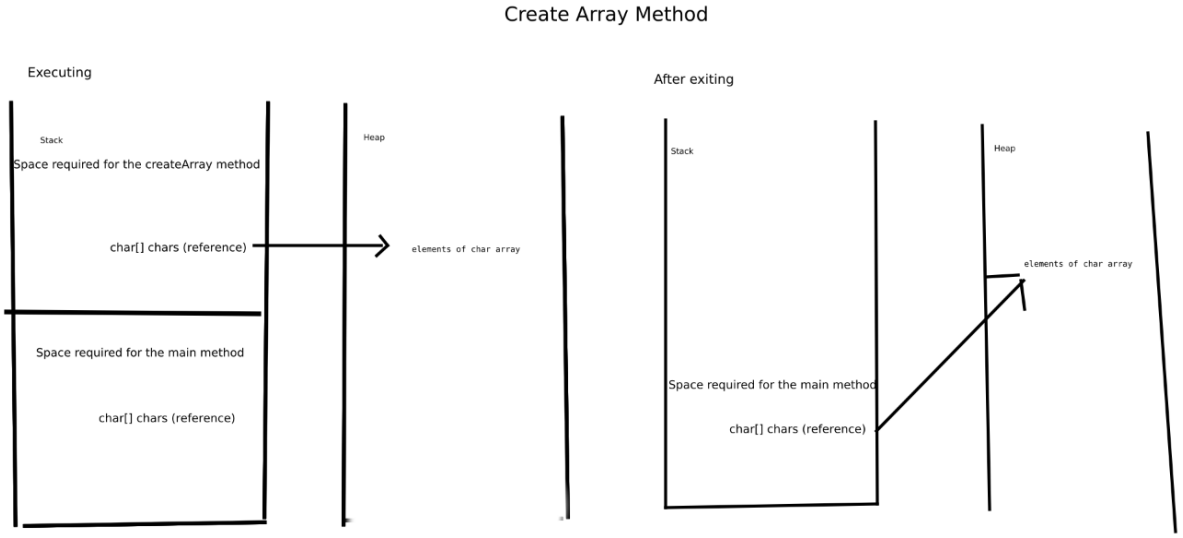
b

a. number is 0 and numbers[0] is 3

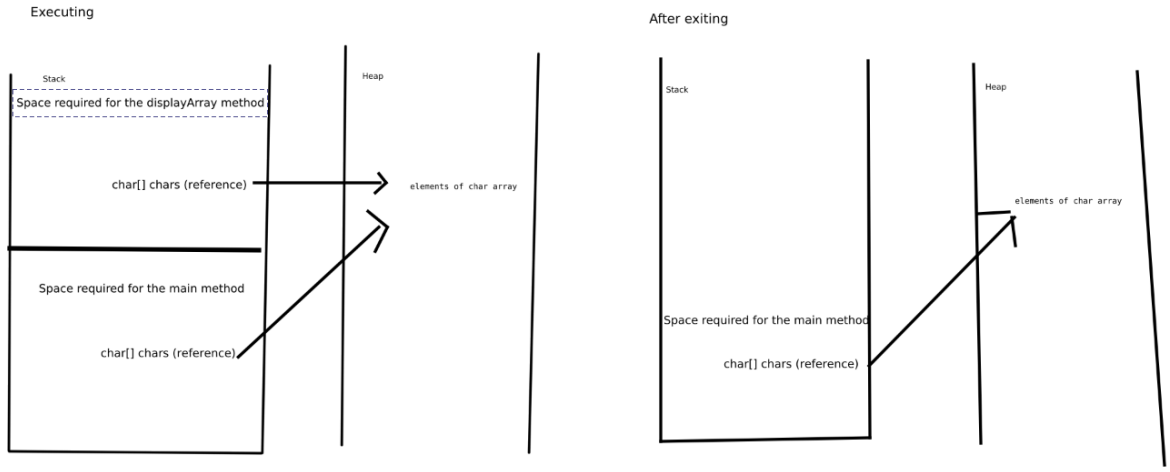
b. 1 2 3 4 5

6.14 | Where are the arrays stored during execution? Show the contents of the stack and heap during and after executing createArray, displayArray , countLetters, displayCounts in Listing 6.4.

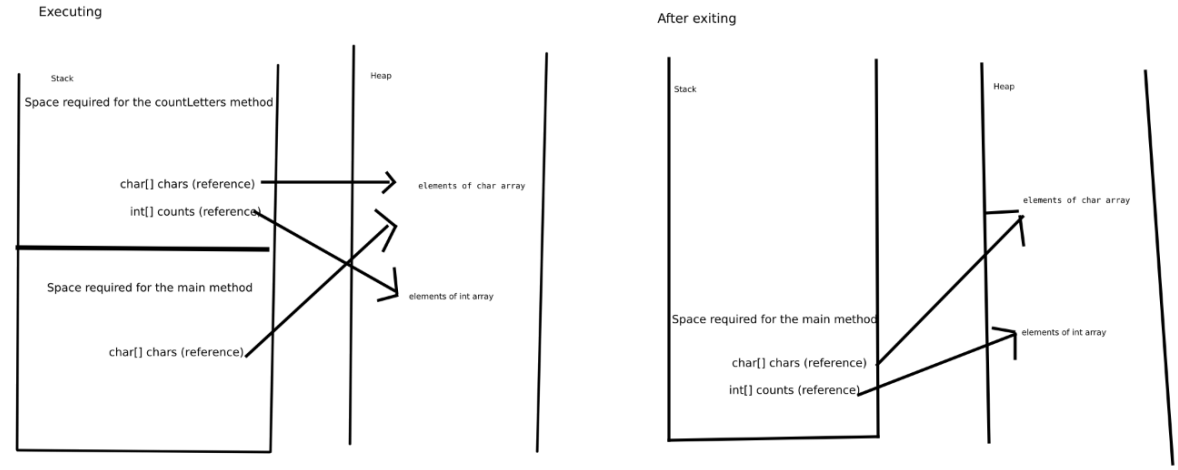
Array's reference variable stores in stack, but array's itself stores in heap.



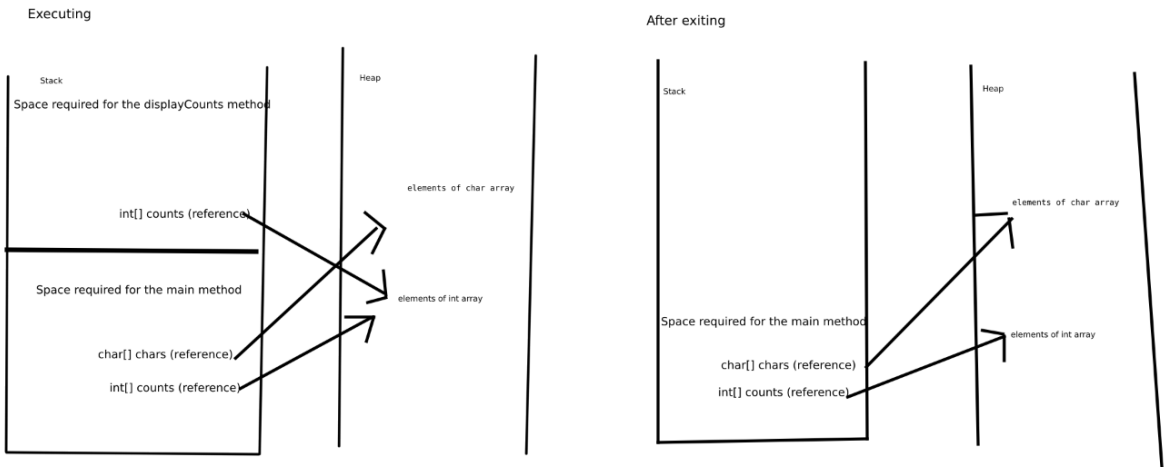
Display Array Method



Count Letters Method



Display Counts Method



Section 6.8

6.15 | What is wrong in the following method declaration?

public static void print(String... strings, double... numbers)

We can’t use more than one variable-length parameter in a method.

public static void print(double... numbers, String name)

We must use variable-length parameter in the last place in method’s argument list

public static double... print(double d1, double d2)

There is no variable-length return type.

6.16 | Can you invoke the printMax method in Listing 6.5 using the following statements?

printMax(1, 2, 2, 1, 4);
printMax(new double[]{1, 2, 3});
printMax(new int[]{1, 2, 3});

You can’t invoke only with the last one. Because the argument array must be double.

6.17 | Use Figure 6.9 as an example to show how to apply the binary search approach to a search for key 10 and key 12 in list 2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 796.

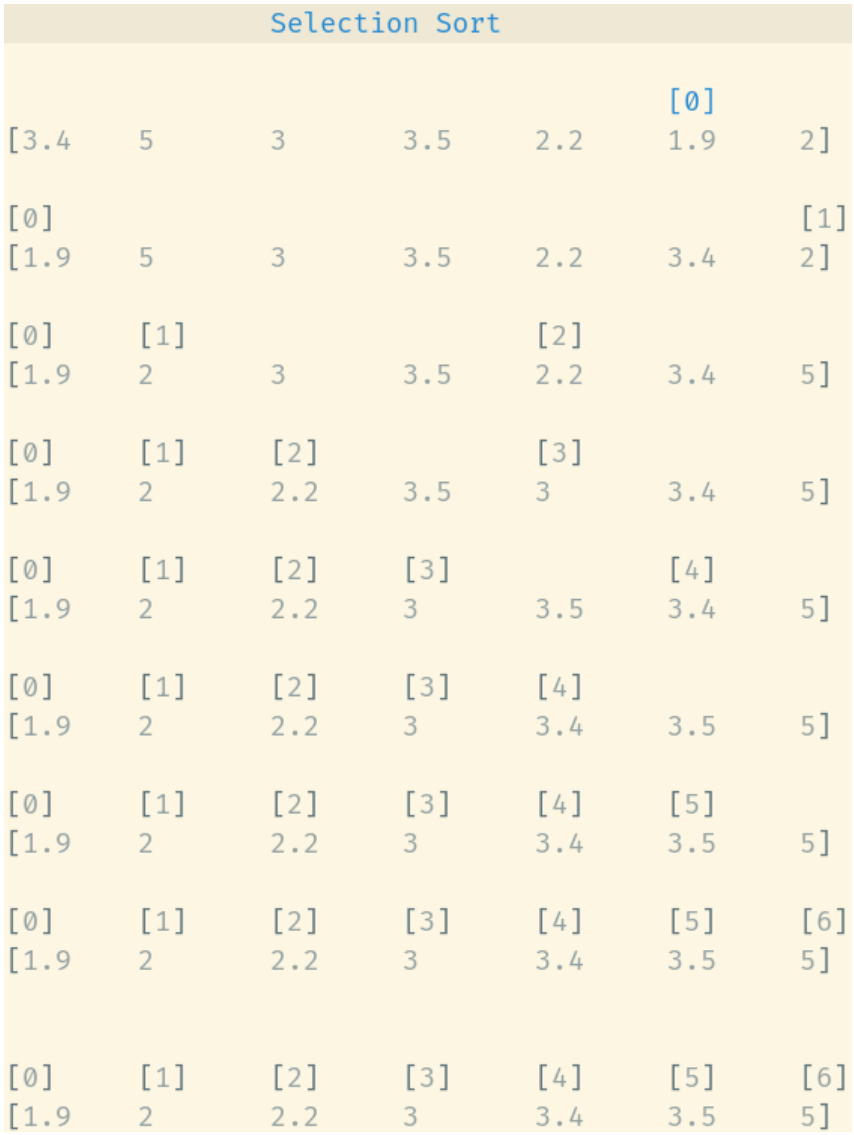
Key = 10

```
low                                    mid                                    high
[2      4       7       10      11      45      50      59      60      66      69      70      79]

low             mid                     high
[2      4       7       10      11      45]

low     mid     high
[10     11      45]

mid
[10]
```

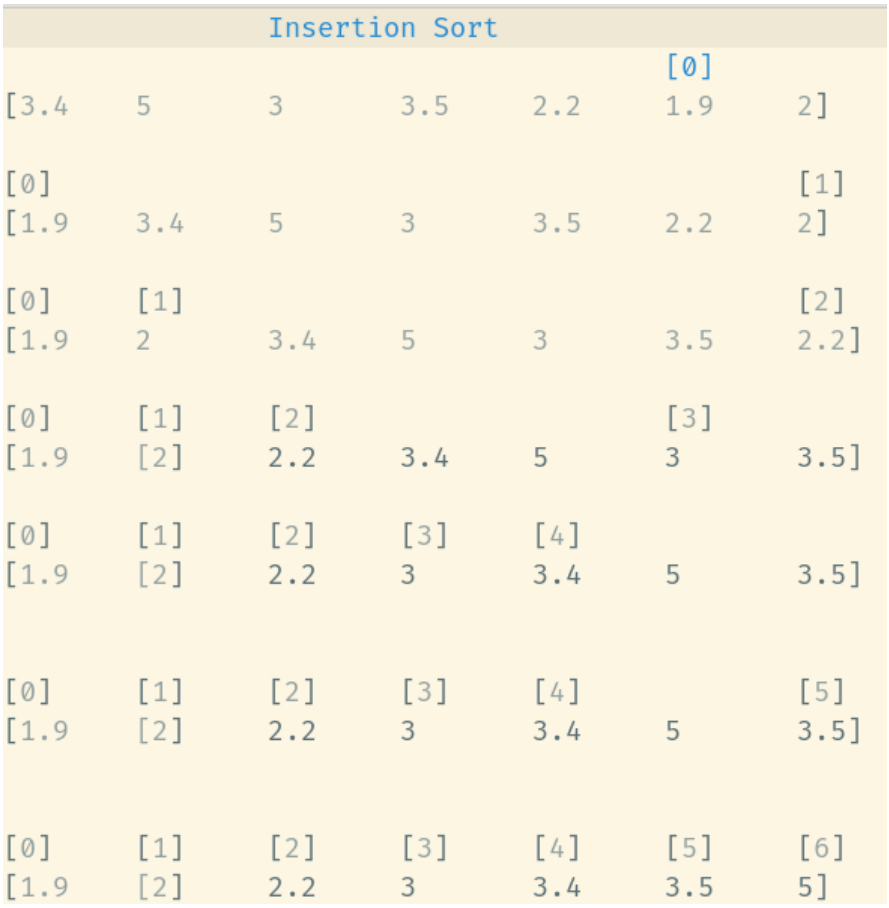
Key = 12

```
low                                    mid                                    high
[2      4       7       10      11      45      50      59      60      66      69      70      79]

low             mid                     high
[2      4       7       10      11      45]

low     mid     high
[10     11      45]

low     high
[11     45]

-1
```

6.18 | Use Figure 6.11 as an example to show how to apply the selection-sort approach to sort { 3.4, 5, 3, 3.5, 2.2, 1.9, 2 }.



6.19 | Use Figure 6.12 as an example to show how to apply the insertion-sort approach to sort { 3.4, 5, 3, 3.5, 2.2, 1.9, 2 }.



6.20 | How do you modify the selectionSort method in Listing 6.8 to sort numbers in decreasing order?

We can solve this problem with just change comparison operator “<” to “>”.

```
for (int j = i + 1; j < list.length; j++)
{
    if (currentMin < list[j])
    {
        currentMin = list[j];
        currentMinIndex = j;
    }
}
```

```
for (int j = i + 1; j < list.length; j++)
{
    if (currentMax > list[j])
    {
        currentMax = list[j];
        currentMaxIndex = j;
    }
}
```

6.20 | How do you modify the insertionSort method in Listing 6.9 to sort numbers in decreasing order?

We can solve this problem with just change comparison operator “>” to “<”.

```
for (k = i - 1; k >= 0 && list[k] > currentElement; k--)
```

```
for (k = i - 1; k >= 0 && list[k] < currentElement; k--)
```

Section 6.11

6.22 | What types of array can be sorted using the java.util.Arrays.sort method?
Does this sort method create a new array?

We can sort whole array or a just a part of an array with using this method. You can sort all of primitive types with this method except boolean.

This method doesn’t create a new array. We use the same array that we use as an argument.

6.23 | To apply java.util.Arrays.binarySearch(array, key), should the array be sorted in increasing order, in decreasing order, or neither?

To be able to using this method, our array must have sorted in increasing order.

6.24 | Show the contents of the array after the execution of each line.

After int[] list = {2, 4, 7, 10};

2
4
7
10

After java.util.Arrays.fill(list, 7);

7
7
7
7

After java.util.Arrays.fill(list, 1, 3, 8);

7
8
8
7

true
After System.out.print(java.util.Arrays.equals(list, list));

7
8
8
7