South Louisiana Community College
ASDV 1220, Programming Fundamentals

Learning Objectives

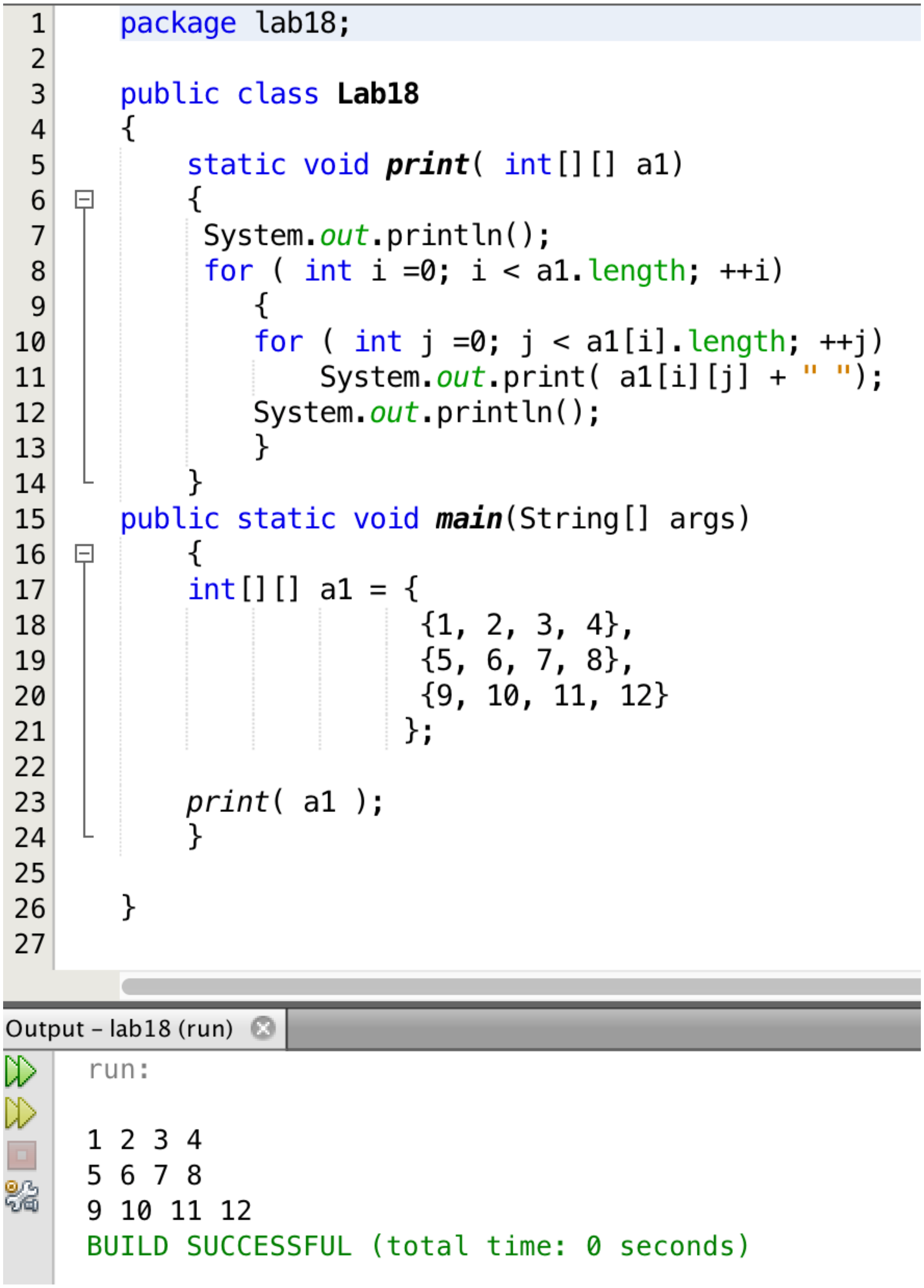
After completion of this lab, you should be able to

1. Understand 2D arrays as parameters and arguments of methods.
2. Understand 2D arrays as return types.
3. Understand array sorting and traversing.

Create project Lab18

Problem 1

Add the following code that creates and initializes a 2D array *a1* Then, call the method *print* to print the contents of the array. Understand how we can traverse a 2D array using loops and indexes where the 1st index is visualized as a row of a table and the second index as a column of a table. Understand the passing of 2D arrays as arguments to methods.



```java
package lab18;

public class Lab18
{
    static void print( int[][] a1)
    {
     System.out.println();
     for ( int i =0; i < a1.length; ++i)
        {
        for ( int j =0; j < a1[i].length; ++j)
            System.out.print( a1[i][j] + " ");
        System.out.println();
        }
    }
public static void main(String[] args)
    {
    int[][] a1 = {
                   {1, 2, 3, 4},
                   {5, 6, 7, 8},
                   {9, 10, 11, 12}
                  };

    print( a1 );
    }

}
```

Output - lab18 (run)

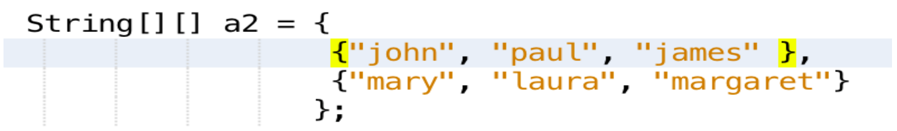
```
run:

1 2 3 4
5 6 7 8
9 10 11 12
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problem 2

Overload method *print* with the following signature:

static void print(String[][] a1)

The code inside *print* should be identical with the previous method *print*.

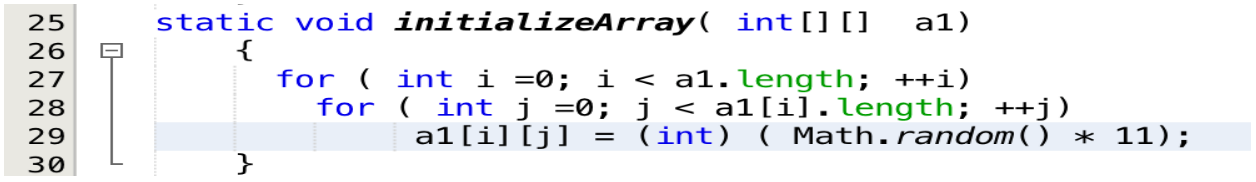


```
String[][] a2 = {
                 {"john", "paul", "james" },
                 {"mary", "laura", "margaret"}
                };
```

Inside your main, create and initialize a 2D called a2 with the following values, then call the *print* method.

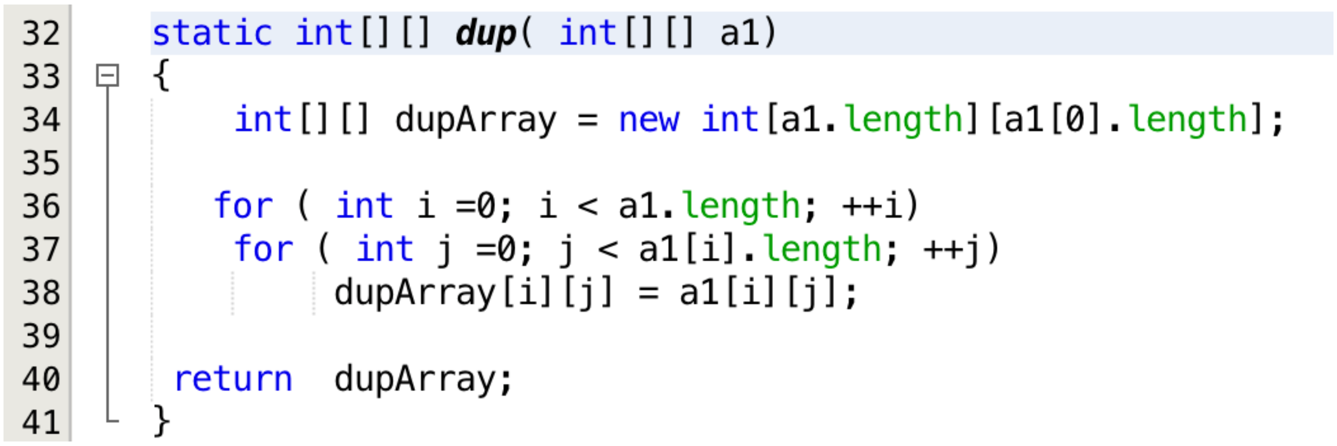
Problem 3

Add the following method that initializes the parameter array a1 to random values between 0 and 10. Inside your main method, create a 5X5 array called *a3*. Pass array *a3* to method *initializeArray* and then call the *print* method to print the initialized array.



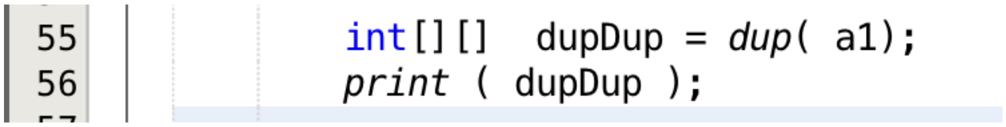
```
    static void initializeArray( int[][]  a1)
        {
          for ( int i =0; i < a1.length; ++i)
            for ( int j =0; j < a1[i].length; ++j)
                 a1[i][j] = (int) ( Math.random() * 11);
        }
```

Problem 4
Add the following method that duplicates the contents of the parameter array and returns the duplicated array as a return-type.



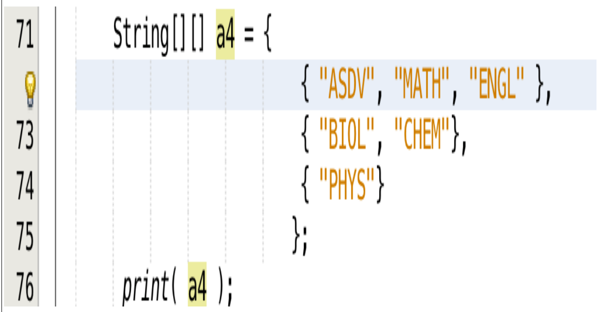
```java
    static int[][] dup( int[][] a1)
    {
        int[][] dupArray = new int[a1.length][a1[0].length];

       for ( int i =0; i < a1.length; ++i)
        for ( int j =0; j < a1[i].length; ++j)
            dupArray[i][j] = a1[i][j];

     return  dupArray;
    }
```

Call the method *dup* from main to duplicate your integer array. Then, print the duplicate array as shown below.

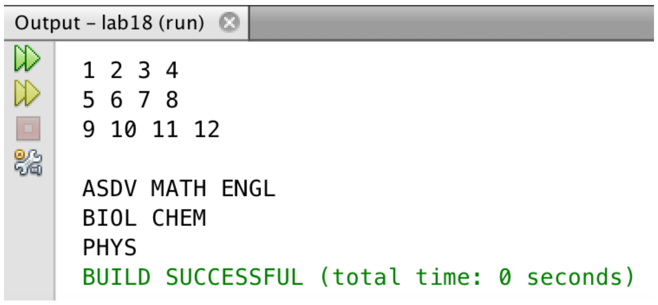


```java
        int[][]  dupDup = dup( a1);
        print ( dupDup );
```

Problem 5
Inside *main*, add the following method that creates a jagged 2D array called *a4* and then call the method *print* to print the jagged array.



```java
    String[][] a4 = {
                    { "ASDV", "MATH", "ENGL" },
                    { "BIOL", "CHEM"},
                    { "PHYS"}
                    };
     print( a4 );
```



Output – lab18 (run)

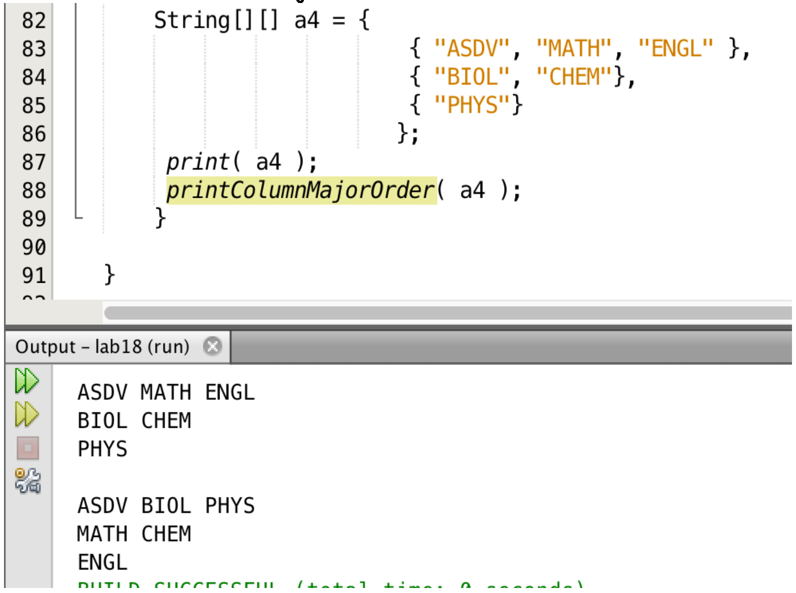
```
1 2 3 4
5 6 7 8
9 10 11 12

ASDV MATH ENGL
BIOL CHEM
PHYS
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problem 6

Add a following method according to the header:

static void printColumnMajorOrder(String[][] a1)

The method *printColumnMajorOrder* prints the 2D array in column-major order. That is, prints down columns and not across rows as the previous print methods. Test it with the jagged array. The print outs for row-major-order and column-major-order are shown below:



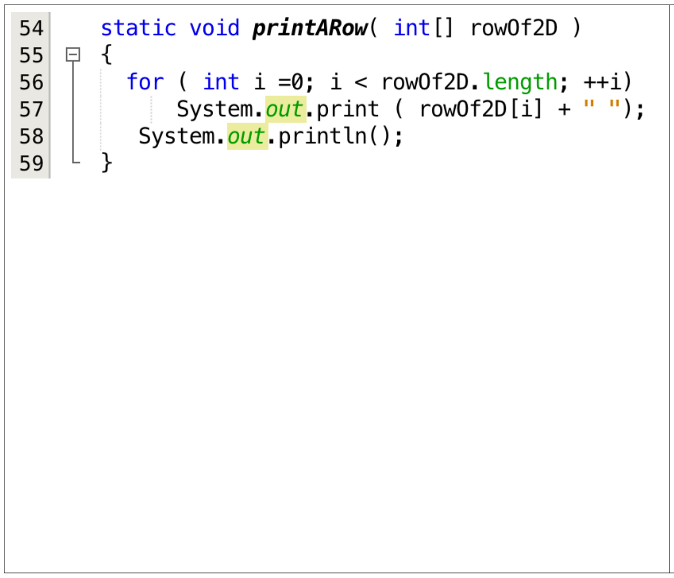
```
        String[][] a4 = {
                            { "ASDV", "MATH", "ENGL" },
                            { "BIOL", "CHEM"},
                            { "PHYS"}
                          };
         print( a4 );
         printColumnMajorOrder( a4 );
        }

    }
```

Output – lab18 (run)

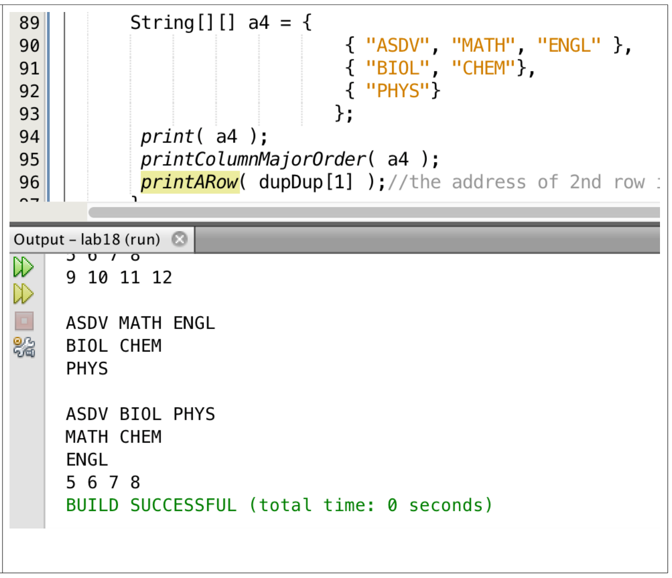
```
ASDV MATH ENGL
BIOL CHEM
PHYS

ASDV BIOL PHYS
MATH CHEM
ENGL
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problem 7

Add a following method that prints an 1D array. Then, call it as shown in line 96 below to print the second row of the 2D array *dupDup*. Observe that in line 96 we pass to method *printArow* as argument the ADDRESS of the second row of the 2D array dupDup.



```
54      static void printARow( int[] rowOf2D )
55      {
56        for ( int i =0; i < rowOf2D.length; ++i)
57            System.out.print ( rowOf2D[i] + " ");
58         System.out.println();
59      }
```



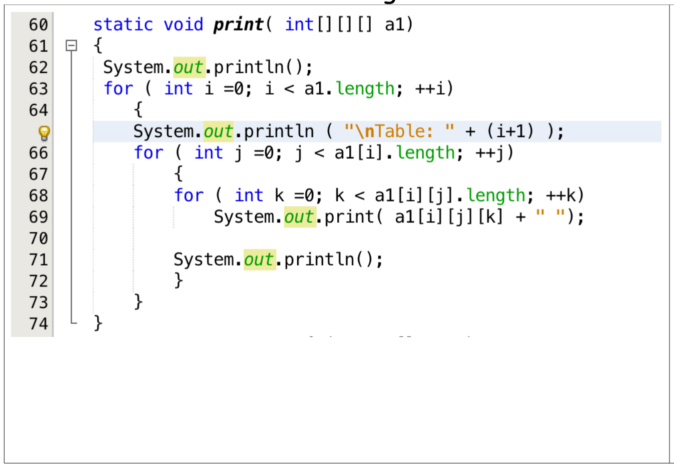
```
89          String[][] a4 = {
90                              { "ASDV", "MATH", "ENGL" },
91                              { "BIOL", "CHEM"},
92                              { "PHYS"}
93                            };
94           print( a4 );
95           printColumnMajorOrder( a4 );
96           printARow( dupDup[1] );//the address of 2nd row
```

Output – lab18 (run)

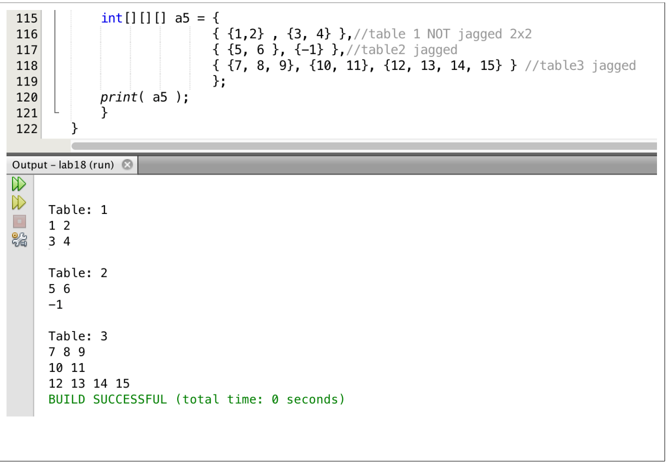
```
5 6 7 8
9 10 11 12

ASDV MATH ENGL
BIOL CHEM
PHYS

ASDV BIOL PHYS
MATH CHEM
ENGL
5 6 7 8
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problem 8

Overload method *print* that prints a 3D array. Observe the testing-condition of the length of the array and understand lines 63, 66 and 68. This method prints any 3D array of any size, jagged or not. Test it as shown on the right hand side.



```
60      static void print( int[][][] a1)
61      {
62       System.out.println();
63       for ( int i =0; i < a1.length; ++i)
64          {
        System.out.println ( "\nTable: " + (i+1) );
66          for ( int j =0; j < a1[i].length; ++j)
67              {
68              for ( int k =0; k < a1[i][j].length; ++k)
69                  System.out.print( a1[i][j][k] + " ");
70
71              System.out.println();
72              }
73          }
74      }
```

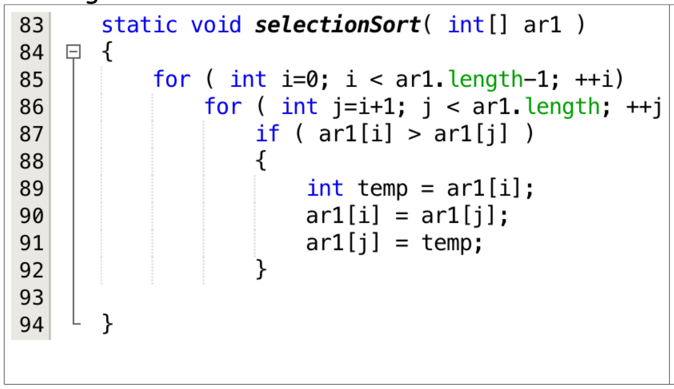


```
        int[][][] a5 = {
                        { {1,2} , {3, 4} },//table 1 NOT jagged 2x2
                        { {5, 6 }, {-1} },//table2 jagged
                        { {7, 8, 9}, {10, 11}, {12, 13, 14, 15} } //table3 jagged
                        };
        print( a5 );
        }
    }
```

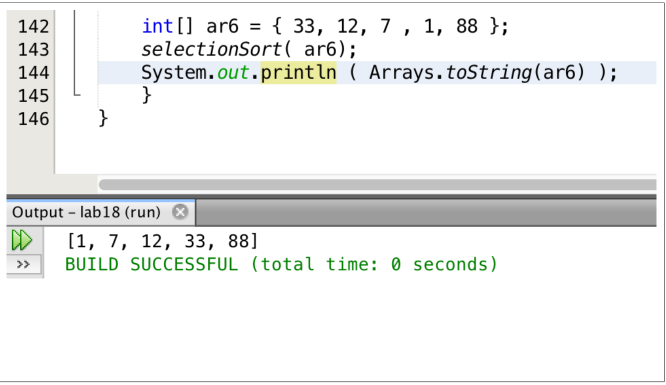
```
Table: 1
1 2
3 4

Table: 2
5 6
-1

Table: 3
7 8 9
10 11
12 13 14 15
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problem 9

Add the following method *selectionSort* that sorts in int array into ascending order. Undestand how it works from your notes in class or by using the Debugger at each step. Test its as shown on the right hand side.



```
static void selectionSort( int[] ar1 )
{
    for ( int i=0; i < ar1.length-1; ++i)
        for ( int j=i+1; j < ar1.length; ++j
            if ( ar1[i] > ar1[j] )
            {
                int temp = ar1[i];
                ar1[i] = ar1[j];
                ar1[j] = temp;
            }

}
```

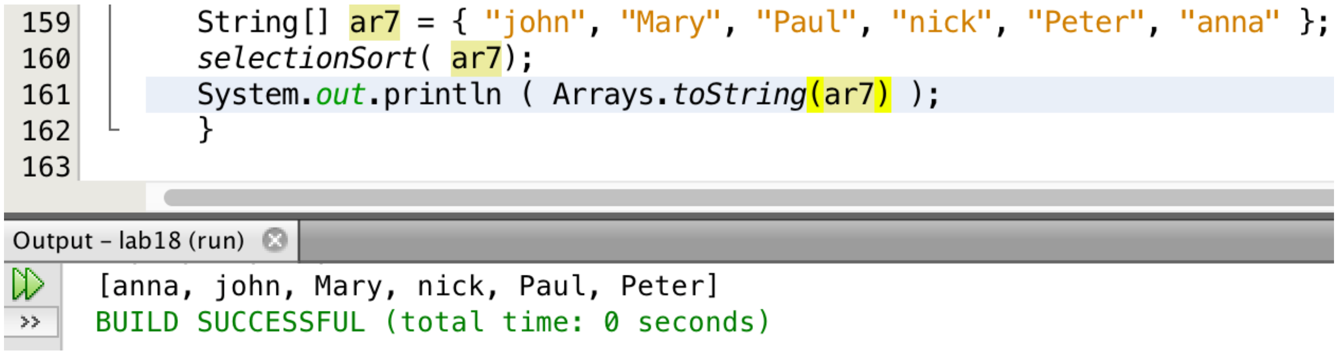


```
        int[] ar6 = { 33, 12, 7 , 1, 88 };
        selectionSort( ar6);
        System.out.println ( Arrays.toString(ar6) );
        }
    }
```

```
[1, 7, 12, 33, 88]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problem 10

Overload method selectionSort *to sort an array of String type.* Test it with the following code. The comparison is NOT case sensitive.



```
        String[] ar7 = { "john", "Mary", "Paul", "nick", "Peter", "anna" };
        selectionSort( ar7);
        System.out.println ( Arrays.toString(ar7) );
        }
```

```
[anna, john, Mary, nick, Paul, Peter]
BUILD SUCCESSFUL (total time: 0 seconds)
```