# CSE 240
# Homework 1 – Programming with Java
# Due: Wednesday, August 29, 11:59 PM

## 1. What This Assignment Is About:

- Classes (methods and attributes)
- Objects
- Arrays of Primitive Values
- Arrays of Objects
- Recursion
- for and if Statements
- Selection Sort

## 2. Use the following Guidelines

- Give identifiers semantic meaning and make them easy to read (examples numStudents, grossPay, etc).
- User upper case for constants. Use title case (first letter is upper case) for classes. Use lower case with uppercase word separators for all other identifiers (variables, methods, objects).
- Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes classes, methods, and code associated with ifs, switches and loops. Be consistent with the number of spaces or tabs that you use to indent.
- Use white space to make your program more readable.

**For each file (class) in your assignment, provide a heading (in comments) which includes:**

- **The assignment number.**
- **Its author (your name).**
- **A description of what this program is doing.**

## 3. Part 1. Primitive Types, Searching, Recursion (35 points).

a) Create a class **Homework** (in a file Homework.java)

b) Create a **static** method **initializeArray** that receives as a parameter an array of integers. Use a for loop and an if statement to put 1s in the odd positions of the array and 0s in the even positions.

c) Create a **static** method **printArray** that receives as a parameter an array of integers. Use a for statements to print all the elements in the array.

d) Create a **static** method **selectionSort** that receives as a parameter an array of integers and order its element in descending order. Implement Selection Sort algorithm. It should be Selection Sort, not Bubble Sort, not Quick Sort, etc. If you do not remember selection sort, this link could be useful: https://goo.gl/hrAdMo

e) Create a **static** recursive method that calculate and returns the **factorial** a number. The method receives the number (integer number) as parameter

f) Copy the following main method in your class,

```
public static void main (String [] arg) {
  int [] a = {3, 5, 6, 8, 12, 13, 16, 17, 18, 20};
  int [] b = {18, 16, 19, 3 ,14, 6};
  int [] c = {5, 2, 4, 3, 1};

  // testing initializeArray
  printArray(a); // print: 3, 5, 6, 8, 12, 13, 16, 17, 18, 20
  initializeArray(a);
  printArray(a); // print: 0, 1, 0, 1, 0, 1, 0, 1, 0, 1

  // testing initializeArray
  printArray(b); // print: 18, 16, 19, 3 ,14, 6
  selectionSort (b);
  printArray(b); // print: 19, 18, 16, 14, 6, 3

  // testing factorial
  System.out.println (factorial (5)); //print: 120

  c[0] = factorial (c[0]);
  c[1] = factorial (c[2]);
  printArray(c); // print: 120, 24, 4, 3, 1
}
```

## Grading Criteria for the part 1

01 pts: file contains header information
01 pts: adequate comment to explain every method
01 pts: consistent indentation and spacing
08 pts: selectionSort
08 pts: printArray
08 pts: initializeArray
08 pts: factorial

## 4. Part 2 Classes, Objects, and Arrays (65 points).

In this assignment, we will be making a program that reads in patrons' information and create a movie theatre seating with a number of rows and columns specified by a user. Then it will attempt to assign each patron to a seat in a theatre.

Use the file Assignment.java (attached at the end of this document). Do not change the content of this file.

Save all files in the same folder.

**Step 1.**

First, you need to create Patron.java file by defining Patron class. It should have two instance variables, lastName (String) and firstName (String). In addition, the following methods should be defined.

| **Method** | **Description of the Method** |
|---|---|
| public Patron ( ) | Constructs a Patron object by assigning the default string " *** " to both instance variables, lastName and firstName. |
| public Patron (String patronInfo) | Constructs a Patron object using the string containing patron's info. Use the split method of the String class to extract first name and last name, then assign them to each instance variable of the Patron class. An example of the input string is:<br>David/Johnson |
| public String getLastName ( ) | It should return the instance variable lastName. |
| public String getFirstName ( ) | It should return the instance variable firstName. |
| public String toString ( ) | It should constructor a string containing the initial character of the first name, a period, the initial character of the last name, and a period, then it returns it. An example of such string for the patron David Johnson is:<br>*D.J.* |

**Step 2.**

You will be creating a class called TheatreSeating. This class should be defined in a file named "TheatreSeating.java". The class TheatreSeating will contain a 2-dimensional array called "seating" of Patron objects at its instance variable. The class TheatreSeating **must** include the following constructor and methods. (If your class does not contain any of the following methods, points will be deducted.)

| **Method** | **Description of the Method** |
|---|---|
| public<br>TheatreSeating<br>(int rowNum, int columnNum) | It instantiates a two-dimensional array of the size "rowNum" by "columnNum" specified by the parameters. Then it initializes each patron element of this array using the constructor of the class Patron without any parameter. So, each patron will have default values for its instance variables. |
| private Patron getPatronAt<br>(int row, intcol) | It returns a patron at the indexes row and col (specified by the parameters of this method) of the array "seating". |
| public boolean<br>assignPatronAt<br>(int row, int col, Patron tempPatron) | The method attempts to assign the "tempPatron" to the seat at "row" and "col" (specified by the parameters of this method). If the seat has a default patron, i.e., a patron with the last name "***" and the first name "***", then we can assign the new patron "tempPatron" to that seat and the method returns true. Otherwise, this seat is considered to be taken by someone else, the method does not assign the patron and returns false. |
| public boolean checkBoundaries<br>(introw, int col) | The method checks if the parameters row and col are valid. If at least one of the parameters "row" or "col" is less than 0 or larger than the last index of the array (note that the number of rows and columns can be different), then it returns false. Otherwise it returns true. |
| public String toString( ) | Returns a String containing information of the "seating". It should show the list of patrons assigned to the seating using the toString method of the class Patron (it shows initials of each patron) and the following format:<br>The current seating<br>--------------------<br>D.J. *.*. E.T..<br>*.*. *.*. S.W.<br>T.C. A.T. *.*.<br><br>Please see the sample output listed below. |

After compiling Patron.java, TheatreSeating.java, and Assignment.java files, you need to execute Assignment.class.

**Sample Output: (the inputs entered by a user are shown in bold)**

Make sure that your program works at least with this scenario.

C:\MyJava\applications>java Assignment

```
Please enter a number of rows for a theatre seating.
3
Please enter a number of columns for a theatre seating.
3
Please enter a patron information or enter "Q" to quit.
Mickey/Mouse
A patron information is read.
Mickey/Mouse
Please enter a row number where the patron wants to sit.
1
Please enter a column number where the patron wants to sit.
2
The seat at row 1 and column 2 is assigned to the patron M.M.

The current seating
--------------------
*.*. *.*. *.*.
*.*. *.*. M.M.
*.*. *.*. *.*.

Please enter a patron information or enter "Q" to quit.
Daisy/Duck

A patron information is read.
Daisy/Duck
Please enter a row number where the patron wants to sit.
2
Please enter a column number where the patron wants to sit.
0

The seat at row 2 and column 0 is assigned to the patron D.D.

The current seating
--------------------
*.*. *.*. *.*.
*.*. *.*. M.M.
D.D. *.*. *.*.

Please enter a patron information or enter "Q" to quit.
Clarabelle/Cow

A patron information is read.
Clarabelle/Cow

Please enter a row number where the patron wants to sit.
2
Please enter a column number where the patron wants to sit.
1

The seat at row 2 and column 1 is assigned to the patron C.C.

The current seating
--------------------
*.*. *.*. *.*.
*.*. *.*. M.M.
D.D. C.C. *.*.
```

```
Please enter a patron information or enter "Q" to quit.
Max/Goof
A patron information is read.
Max/Goof
Please enter a row number where the patron wants to sit.
0
Please enter a column number where the patron wants to sit.
0

The seat at row 0 and column 0 is assigned to the patron M.G.

The current seating
--------------------
M.G. *.*. *.*.
*.*. *.*. M.M.
D.D. C.C. *.*.

Please enter a patron information or enter "Q" to quit.
Horace/Horsecollar
A patron information is read.
Horace/Horsecollar

Please enter a row number where the patron wants to sit.
5
Please enter a column number where the patron wants to sit.
1
row or column number is not valid.
A patron Horace Horsecollar is not assigned a seat.
Please enter a patron information or enter "Q" to quit.
Sylvester/Shyster

A patron information is read.
Sylvester/Shyster
Please enter a row number where the patron wants to sit.
2
Please enter a column number where the patron wants to sit.
0
The seat at row 2 and column 0 is taken.
Please enter a patron information or enter "Q" to quit.
Snow/White

A patron information is read.
Snow/White
Please enter a row number where the patron wants to sit.
-1

Please enter a column number where the patron wants to sit.
0

row or column number is not valid.
A patron Snow White is not assigned a seat.
Please enter a patron information or enter "Q" to quit.
Jiminy/Criket

A patron information is read.
Jiminy/Criket
Please enter a row number where the patron wants to sit.
0
Please enter a column number where the patron wants to sit.
2

The seat at row 0 and column 2 is assigned to the patron J.C.

The current seating
```

```
--------------------
M.G. *.*. J.C.
*.*. *.*. M.M.
D.D. C.C. *.*.

Please enter a patron information or enter "Q" to quit.
Q
```

**Grading Criteria for the part 2**

05 pts: Every file contains header information
05 pts: adequate comment to explain every method
05 pts: consistent indentation and spacing
05 pts: it compiles
05 pts: Two constructors of Patron are correct
05 pts: Accessor methods for lastName and firstName of Patron are correct
05 pts: toString method of Patron is correct
05 pts: Constructor, TheatreSeating(int,int) is correct
05 pts: getPatronAt(int,int) method is correct
10 pts: assignPatronAt(int,int,Patron) method is correct
05 pts: checkBoundaries(int,int) method is correct
05 pts: toString method is correct

```
public class Assignment {
  public static void main(String[] args) {
    TheatreSeating theatreSeating;
    Patron tempPatron;
    int row, col, rowNum, columnNum;
    String patronInfo;
    Scanner scan = new Scanner(System.in);
    // Ask a user to enter a number of rows for a theatre seating
    System.out.println
    ("Please enter a number of rows for a theatre seating.");
    rowNum = scan.nextInt();

    // Ask a user to enter a number of columns for a theatre seating
    System.out.println
    ("Please enter a number of columns for a theatre seating.");
    columnNum = scan.nextInt();

    // instantiate a TheatreSeating object
    theatreSeating = new TheatreSeating(rowNum, columnNum);

    System.out.println
    ("Please enter a patron information or enter \"Q\" to quit.");

    /*** reading a patron's information ***/
    patronInfo = scan.next();

    /* we will read line by line **/
    while (!patronInfo.equalsIgnoreCase("Q")){
      System.out.println("\nA patron information is read.");
      // printing information read from a file.
      System.out.println(patronInfo);

      // creating a patron object using the patron information from a user
      tempPatron = new Patron(patronInfo);

      // Ask a user to decide where to seat a patron by asking
      // for row and column of a seat
      System.out.println
      ("Please enter a row number where the patron wants to sit."); row =
      scan.nextInt();

      System.out.println
      ("Please enter a column number where the patron wants to sit.");
      col = scan.nextInt();

        // Checking if the row number and column number are valid
               // (exist in the theatre that we created.)
      if (theatreSeating.checkBoundaries(row, col) == false) {
        System.out.println("\nrow or column number is not valid.");
        System.out.println ("A patron " + tempPatron.getFirstName() +
          " " + tempPatron.getLastName() + " is not assigned a seat.");
      } else {
        // Assigning a seat for a patron
        if (theatreSeating.assignPatronAt(row,col,tempPatron) == true){
          System.out.println("\nThe seat at row " + row +" and column " +
          col + " is assigned to the patron " + tempPatron.toString());
          System.out.println(theatreSeating);
        } else {
          System.out.println("\nThe seat at row " + row + " and column " +
          col + " is taken.");
        }
      }
      // Read the next patronInfo
      System.out.println
        ("Please enter a patron information or enter \"Q\" to quit.");
      /*** reading a patron's information ***/
      patronInfo = scan.next();
    }
  }
}
```