Java Arrays
LAB 6

1. Objectives:

- Manipulating 1-Dimensional Arrays
- Manipulating 2-Dimensional Arrays
- Introduction to image processing using Array (Image Rotation)

2. Lab Practice:

- Create a new Java Project with Eclipse **: "LabArray"**
- Add a new **TestArrays** program. The program creates an array of 10 integers.

```
public static void main(String[] args) {
      //Create an array of 10 integers
      int arrayOfInt[] = new int[10];
      ....
}
```

- What's the output of following code?

```
 System.out.println(arrayOfInt.length);
```

- What's the output of following code?

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

- What's the output of following code?

```
System.out.println(arrayOfInt[6]);
```

- What's the output of following code?

```
System.out.println(arrayOfInt[10]);
```

- catch the exception - using try and **catch keywords**

```
            try {
                  System.out.println(arrayOfInt[10]);
          } catch(ArrayIndexOutOfBoundsException error) {
            System.out.println("ArrayIndexOutOfBoundsException 
cached");
            error.printStackTrace();

          }
```

- Using a "for" loop, initialize the array with random integer values between 0 and 100. Please have a look at Java API documentation to learn how to use the Class Random: http://docs.oracle.com/javase/7/docs/api/java/util/Random.html

- Create a "`print(int[] arrayOfInt)`" method in the Class **TestArrays** that prints an array of integer on the standard output and test it.

```java
/**
 * Print an array of integers
 * on the standard output.
 * Format: [value0, value1, ..., valueN]
 *
 * @param arrayOfInt the specified array of integers
 */

public static void print(int[] arrayOfInt){
...
}
```

- Create a “`getMaxValue(int[] arrayOfInt)`” method in the Class **TestArrays** that returns the maximum value of an array of integers. Test it.

```java
/**
 * Returns the maximum value
 * of the specified array of integers
 *
 *  @param arrayOfInt the specified array of integers
 */
public static int getMaxValue(int[] arrayOfInt){
...
}
```

- Create a “`getMinValue(int[] arrayOfInt)`” method in the Class **TestArrays** that returns the minimum value of an array of integers. Test it.

- Create a “`sortAscending(int[] arrayOfInt)`” method in the Class **TestArrays** that sorts an array of integers into ascending order. Test it.

```java
/**
 *
 *  Sorts the specified array of objects
 *  into ascending order
 *
 *  @param arrayOfInt the specified array of integer
 */
public static void sortAscending(int[] arrayOfInt){

...

}
```

- Create a “`sortDescending(int[] arrayOfInt)`” method in the Class **TestArrays** that sorts an array of integers into descending order. Test it.
- Have a look at the Java API documentation for Class Arrays: http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html. What do you see? Would it be useful for this lab?
- Rewrite “`print`”, “`sortAscending`”, “`sortDescending`”, “`getMaxValue`”, “`getMinValue`” using **`java.util.Arrays`**

2.1. Two dimensional arrays

- Add a new **TwoDimArraysDemo.java** file. Create a new two dimensional array of integers.

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

   //A simple 2D matrix of integer
   int matrix2D[][] = new int [3][4];

}
```

- How to get the width and height of the above `matrix2D` element? Please provide the corresponding code snippet.
- Using the Class Random, initialize `matrix2D` with random integer values between 0 and 5.
-
- Provide an implementation of the following `printMatrix2D(int[][] matrix2D)` method and test it on the **`matrix2D`** element :

```
/**
 * Prints a 2-dimensional array of integers (matrix)
 * on the standard output.
 * Format:
 * [[value00, value01, ..., value0N],
 *  [value10, value11, ..., value1N],
 *  .....
 *  [valueM0, valueM1, ..., valueMN]]
 *
 * @param matrix2D the specified 2-dimensional array of integers
 *
 */
public static void printMatrix2D(int[][] matrix2D){

...

}
```

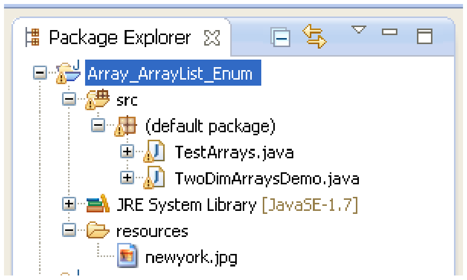
- Propose an implementation of the `rotate90Deg(int[][] matrix2D)` method that performs a clockwise rotation of `matrix2D` and returns the rotated 2D array.

```
/**
 * Performs a 90 degree clockwise rotation of the specified
 * 2D array and returns the rotated 2D array.
 *
 * Before rotation:
 * [[a, b, c],
 *  [d, e, f]]
 *
 * After rotation:
 *  [[d, a],
 *   [e, b],
 *   [f, c]]
 *
 *
 * @param matrix2D the specified 2D array
 * @return the rotated 2D array
 */
public static int[][] rotate90Deg(int[][] matrix2D){...}
```

- Propose an implementation of the `convert2Dto1D(int[][] matrix2D)` method that converts a 2D array into a 1D array. Test it on both the original matrix “`matrix2D`” and the rotated matrix “`rotate90Deg(matrix2D)`”

```
/**
 * Converts a 2D array into 1D array
 *
 * From 2D:
 * [[a, b, c],
 *  [d, e, f]]
 *
 * To 1D:
 * [a, b, c, d, e, f]
 *
 * @param matrix2D the specified 2D matrix
 * @return the 1D matrix derived from the specified 2D matrix
 */
public static int[] convert2Dto1D(int[][] matrix2D){...}
```

- Add a “resources” folder into your project and put an image file (jpg format) in it.



- New -> folder

- The code to manipulate and display an image may use a part of the Java API that is not yet covered by the course. For this lab, just use the methods provided below as is. All details are addressed in future labs.

```
/**
 * Returns a 2D array of pixels
 * from a specified image file
 *
 * @param path the path to the image file
 * @return a 2D array of pixels
 */
public static int[][] getPixels2DFromFile(String path){
      //Get the image from file
      BufferedImage image = null;
      try {
            image = ImageIO.read(new File(path));
      } catch (IOException e) {
            System.err.println("File "+path+" not found.");
      }

      // Fill the 2D array with the image pixels
      int width = image.getWidth(); //number of columns
      int height = image.getHeight(); //number of rows
      int [][] pixelsMatrix = new int[width][height];
      for (int w = 0; w < width; w++){
            for (int h = 0; h < height; h++){
                  pixelsMatrix[w][h] = image.getRGB(w, h);
            }
      }
      return pixelsMatrix;
}
/**
 *
 * Creates an RGB image from the specified 1D array of pixels.
 *
 * @param pixels 1D array of pixels
 * @param width the image width
 * @param height the image height
 * @return a BufferedImage from the specified array of pixels
 *
 */
public static BufferedImage createRGBBufferedImage(
                        int[] pixels, int width, int height){
      //Creates a BufferedImage
      BufferedImage image = new BufferedImage(
                  width, height, BufferedImage.TYPE_INT_RGB);
      //Fills the image with the pixels
      image.setRGB(0, 0, width, height, pixels, 0, width);

      return image;

}
```

```
/**
 * Displays the specified image on screen
 *
 * @param image the specified image
 */
public static void displayImage(BufferedImage image){
      JFrame frame = new JFrame();
      frame.getContentPane().add(new JLabel(new
      ImageIcon(image)));
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.pack();
      frame.setVisible(true);
}
```

- Using the methods provided above, complete the **TwoDimArraysDemo** program to display the picture stored in the *resources* folder.
- Using the methods provided above and the one previously designed, complete the **TwoDimArraysDemo** program to rotate the picture stored in the resources folder and display the rotated image.