**Lesson 02**

**Learning Objectives:**

* Familiarize with the SDLC.
* Use GitHub for assignments.
* Review double arrays and loops.
* Understand how to work with RGB.
* Go to D2L, GitHub and select PP01.
  + Load Assignment
  + Go over VS Code and Loading Project
    - Different Files
    - Problems/TODOs
    - Committing Changes
* **Double arrays review**
  + Syntax: type[][] = new type[num\_rows][num\_cols]
  + Example: int[][] grades = new int[20][5]
    - Say for 20 students each with 5 grades
    - Traversal – TestIM.java print array in repo PP01-Assignment
* Colors – Slides
* ImagesManipulator

public int[][] grayScale() {

        // TODO (in class): average the R G B values

        int w = pixels.length;

        int h = pixels[0].length;

        int[][] newPixels = new int[w][h];

        for (int x = 0; x < w; x++)

            for (int y = 0; y < h; y++) {

                // average the red, blue, and green values

                // of the pixel and set the corresponding

                // grayscale pixel to this average value

                Color color = new Color( this.pixels[x][y]);

                int red = color.getRed();

                int blue = color.getBlue();

                int green = color.getGreen();

                int gray = (red + blue + green) / 3;

                newPixels[x][y] = new Color(gray,gray,gray).getRGB();

            }

        return newPixels;

 }

public int[][] filter(int mask) {

        // TODO (in class): use & to filter by mask

        int w = pixels.length;

        int h = pixels[0].length;

        int[][] newPixels = new int[w][h];

        for (int x = 0; x < w; x++)

            for (int y = 0; y < h; y++) {

                int color = this.pixels[x][y];

                newPixels[x][y] = mask & color;

            }

        return newPixels;

* **The Software Lifecycle:**
  + The development of good software involves a lengthy and continuing process known as the software’s **life cycle**:
    - Specification:
      * Given an initial statement of the software’s purpose, you must specify all aspects of the problem.
      * The specification phase requires that you bring precision and detail to the original problem statement and that you communicate with both programmers and nonprogrammers.
    - Design:
      * Once you have completed the specification phase, you must design a solution to the problem.
      * The best way to simplify the problem-solving process is to divide a large problem into small, manageable parts. The resulting program will contain **modules**, which are self-contained units of code.
      * Classes should be designed so that the objects are independent, or loosely coupled. Coupling is the degree to which objects in a program are interdependent.
      * Classes should also be designed so that objects are highly cohesive. Cohesion is the degree to which the data and methods of an object are related.
      * Methods should also be highly cohesive; each should perform one well-defined task.
      * Ideally, each object should represent one component in the solution.
      * Objects interact by sending messages to each other through method calls, which in turn represents the **data flow** among objects.
        + You should specify in detail the assumptions, input, and output for each method.
        + You can view these specifications as the terms of a **contract** between your method and the code that calls it.
    - Implementation
      * The coding phase involves translating the design into a particular programming language and removing the syntax errors.
    - Testing/Refining
      * During the testing phase, you need to remove as many logical errors as you can. One approach is to test the individual methods of the objects first, using valid input data that leads to a known result.
      * Often the best approach to solving a problem is first to make some simplifying assumptions during the design of the solution and next to develop a complete working program under these assumptions. You can then add more sophisticated input and output routines, additional features, and more error checks to the working program.
      * Also, realize that any time you modify a program—no matter how trivial the changes might seem—you must thoroughly test it again.
    - Production/Deployment
      * When the software product is complete, it is distributed to its intended users, installed on their computers, and used.
    - Maintenance
      * Users of your software invariably will detect errors that you did not discover during the testing phase. Correcting these errors is part of maintaining the software.
      * Another aspect of the maintenance phase involves enhancing the software by adding more features or by modifying existing portions to suit the users better.