Project Description

This *individual or team programming* project is worth 25% of your total grade (100 points for scoring purposes). Team project requirements include election of a team leader who reports member participation.

Item	Points Possible
Part (1) – Design/Documentation	20 points (or 5% of total grade)
Part (2) - Implementation	80 points (or 20% of total grade)

Please note that this project must be unique in its design and implementation. You are not to copy or use any part of a Java *program* that was previously submitted or appears on the Internet, in a textbook, or otherwise made available via an external source. **You *may* however use your original code/design/documentation from your COSC 1436 Java project for this assignment. That is, you may *continue* development of the project that you started in COSC 1436.** Contact your instructor if you have any questions regarding this requirement.

DELIVERABLES:

PART 1: Design/Documentation (20 points). Minimum requirements.

1. **Project Description** (5 points). **Executive Summary**

   Provide a 1/2 to 1-page overview of your project identifying the major components as part of a Java program design. The format is an executive summary to be presented to management for review. Suggested sections include introduction, recommendations, and justification.

   Note: No third-party software function library is permitted.

2. **Structured, valid, efficient, commented code** with a reasonable level of code and data structure content. Demonstrate coding standards presented in the textbook. (5 points).

3. **This form** with all *blanks* filled in. (5 points).

4. **Documentation** (5 points). One or more of the following: diagram (UML, etc.), flow chart, pseudocode, documentation generated via documentation-generator such as *javadoc*, documentation - any format, external, User Guide, specifications, requirements, etc. Note: You can use Microsoft Visio, Word, or Access as a diagramming tool. Please obtain approval regarding the use of other diagramming tools.

PART 2: Implementation (80 points) Minimum Requirements.

1. **Interface(s)** (5 points). One or more of the following:
   a. Graphical User Interface (ex: Java Foundation Classes)
   b. Dialog Boxes (ex: jOptionPane class)
   c. Text (ex: Java API classes: ex.: print & println methods)

   Identify selected interface(s): **C**

2. **Array(s)** (5 points). One or more of the following: primitive variable array, array of objects, parallel arrays, two, or more dimension array, array of objects using ArrayList class.

   Identify array type(s) / name(s) / location(s): **Garden.java, 2D array, line 17**

3. **Class Members** (10 points). One or more of the following. Including any appropriate constructor(s), accessor and mutator methods.
   a. A `static` field (Ref: Ch.8, p.496) or method (Ref: Ch.8, p.499)
   b. A `Character` wrapper class (Ref: Ch.9, p.560) Using one or more of the following methods: `isDigit, isLetter, isLetterOrDigit, isLowerCase, isUpperCase, isSpaceChar, isWhiteSpace, toLowerCase, toUpperCase`
   c. A `Numeric` wrapper class (Ref: Ch.9, p.597) using one or more of the following wrapper classes: `Byte, Double, Float, Integer, Long, Short`

   Name(s) / location(s) of class(es): **Integer.parseInt() used in Garden.java, line 137**

   Names(s) / location(s) of object(s): **Character.toUpperCase() used in Garden.java, line 97**

   Name(s) / location(s) of field(s) or method(s): **Static fields and methods used in Garden.java**

4. **Aggregation of two or more classes** (10 points). (Ref: Ch.8, p.517)

   Name(s) / location(s) of aggregated class(es):

5. **An inherited class (using 'extends' keyword, not from 'imported' classes like JFrame imported from 'javax.swing.*')** (10 points). (Ref: Ch.10, p.613)

   Name(s) / location(s) of superclass(es): **Tile.java**

   Name(s) / location(s) of subclass(es): **Plant.java extends Tile**

6. **Decision Structure** (5 points). One or more of the following: `If, If-else, Nested if, If-else-if, Switch`

   Name(s) / location(s) of method(s) using decision structure(s): **Switch in Garden,java, line 129**

7. **Repetition Structure** (5 points). One or more of the following: `while, do-while, for`

   Name(s) / location(s) of method(s) using repetition structure(s): **For loop in Garden.java, line 302**

8. **Sequential, Binary, or Random File I/O** (5 points total). (Ref: Ch.4, p.230 & Ch.12, p.703 & Appendix H) Including one or more of the following: writing and/or appending output to a file or reading file as input.

   Name(s) of file(s): **scores.txt**

   Name(s) / location(s) of method(s) using file(s): **FileWriter at Garden.java, line 100**

9. ***Two or more* of the following advanced features/methods. (25 points)**
   a. **Advanced GUI component(s).** (Ref: Ch.13, p.849): List, Combo Box, File Chooser, Color Chooser, Menu, Text Area, Slider
   b. **Advanced File I/O. Write catch clauses to handle each type of exception that could potentially be thrown**. Use a try block and one or more catch clause(s). (Ref: Ch. 11, p. 703)
   c. **Applet.** (Ref: Ch.14, p.917)
   d. A **toString** method to indicate an object's state (Ref: Ch.8, p.507)
   e. An **equals** method to compare contents of objects (Ref: Ch.8, p.511)
   f. A method to copy an object (Ref: Ch.8, p.514) *or* a copy constructor (Ref: Ch.8, p.516)
   g. **Math** class field or method. Use a Math class field or method, or the Java API Random class.

      LINK TO MATH CLASS DOCUMENTATION:

      https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

   Name(s) / location(s) of 2 or more feature(s)/method(s)/constructors listed above:

   (1) Advanced File I/O in Garden.java, try-catch block on lines 100 through 109
   (2) Math.abs used in Garden.java, line 137