



### Assignment Overview

A garden is defined here as consisting of plants. Using the provided class *Plant*, write an object-oriented Java program that keeps track of plants in such a garden.

### Assignment



3. Add the sample *Plant* class file provided on Moodle to the new project. Download and save the file, then right click on the saved file and select copy. In IntelliJ, right click on the package (the default is “com.company”) and select paste.

*Tip: Make an object of class Plant and practice calling the constructors and methods in the class.*

4. In a new file located in the same package as the classes *Main* and *Plant*, create a “normal” *public* Java class to represent a garden. When the garden is first created there are no plants, but later it will have plants. The garden can hold at least 3 plants, and it has just been watered. To represent this information, the garden class must have
- (a) three *private* fields as data members:
    - i. an array of *Plant* reference variables. The size of the array is set when the garden class object is created, but the array’s size must be at least 3 regardless of user input.
    - ii. the number of *Plants* currently in the garden, i. e. the number of plant objects currently being referenced by elements of the array. The number of plants referenced in the array always begins at zero.
    - iii. the number of days since the garden was last watered. The garden is assumed to be watered when created, thus the number of days should begin at zero and never be set to a negative value.
  - (b) a parameterized constructor that receives one parameter giving the maximum number of plants, i. e. giving the size of the array of *Plant* class type to be created. The *Plant* array reference variable is assigned to a new *Plant* array object of the size value received, but with at least 3 elements.
  - (c) a default constructor that assumes the maximum number of plants to be 3 and otherwise acts just as the parameterized constructor.
  - (d) a *public* “getter” (but no “setter”) method for the number of plants field. This returns the number of plants stored in the array, not the number of elements in the array.
  - (e) a *public* “getter” and “setter” method for the number of days since watered field.
  - (f) two *public* methods that each add one plant to the garden as long as there is room in the array of plants. *Note that the number of plants field is the location of the next available element in the array.*
    - i. One add plant method receives one parameter, a *Plant* reference variable, and stores the reference in the first available position in the *Plant* array. The number of plants is increased upon successful planting.
    - ii. The other add plant method receives two parameters, the name of the plant and the height of the plant, creates a new *Plant* class object and stores the reference to the new object in the first available position in the *Plant* array. The number of plants is increased upon successful planting.
  - (g) a *public* method that receives a plant number that is used as the index in the plant array and returns a *String* containing the name of the plant, the height of the plant, and whether the plant needs to be watered. If the plant number received is outside of the range of indexes or refers to an element in the array that does not reference a plant, the value *null* is returned.
  - (h) a *public* method to check if any of the *Plant* class objects needs to be watered by checking to see if the return value of each *Plant* object’s *getNeedsWater* method is *true*. If at least one *Plant* object needs to be watered, water the garden by calling the *waterPlant* method on each *Plant* object and set the garden’s number of days since watering field to zero.

5. In class *Main*, use the *Garden* and *Plant* classes to create an application that allows the user to create then manage a virtual garden. Use user input to set the number of plants allowed in the garden object, then, in a loop, present a menu of options for the user to select. The loop should continue until the user elects to stop. The minimum options in the menu are
- show the number of plants in the garden
  - add a plant to the garden by getting its name and height from the user
  - add a plant to the garden by giving a pre-defined object. You may use a call to one of the *static* “factory” methods to create a plant object:
    - `createFlower()`
    - `createShrub()`
    - `createTree()`
  - increase the number of days since the garden has been watered by one
  - water the garden if at least one plant needs water
  - show the information about a specific plant in the garden
  - show the information about all plants in the garden
  - halt the application

### **Additional Requirements**

- (a) The *Plant* class may not be changed in any way except for the *package* statement (if necessary).
- (b) There is no user I/O (input / output) in the garden class. All values given to the fields are either initial values or values passed to parametrized constructors or setter methods. Values of the fields are returned from the class via methods.
- (c) A reference variable and instance object of class *java.util.Scanner* must be used to read user input.
- (d) Identifiers must be descriptive, i. e. must self document.
- (e) Indention of all code blocks (compound statements, anything in braces), including single statements following selection or while statements, is required.

### **Submitting**

In IntelliJ, select File, Save All, then select File, Export to Zip File, navigate to a location you can find (but DO NOT put the zip inside the project folder), then click OK to save the Zip archive file. Finally, upload the zip archive file to Moodle.

*Helpful Hint: Keep a backup copy of your project folder on a Google Drive, a Drop Box Account, a USB memory device, etc., or even on Moodle! Finally, once you turn in your final version, create a copy before the due date and do not change this copy in any way. This final copy can be consulted if there is an upload disaster, but only if the “.java” files have not been changed in any way after the due date.*