Unit 8 FRQ – 2D Arrays and `ExperimentalFarm`

Haidian Foreign Language Tengfei School – Chris Nielsen

Free Response Question 1

The `ExperimentalFarm` class represents crops grown on an experimental farm. An experimental farm is a rectangular tract of land that is divided into a grid of equal-sized plots. Each plot in the grid contains one type of crop. The crop yield of each plot is measured in bushels per acre.

A farm plot is represented by the `Plot` class. A partial definition of the `Plot` class is shown below.

```
public class Plot
{
    private String cropType;
    private int cropYield;

    public Plot(String crop, int yield)
    {
        /* implementation not shown */
    }

    public String getCropType()
    {
        return cropType;
    }

    public int getCropYield()
    {
        return cropYield;
    }
}
```

The grid of equal-sized plots is represented by a two-dimensional array of `Plot` objects named `farmPlots`, declared in the `ExperimentalFarm` class. A partial definition of the `ExperimentalFarm` class is shown below.

```
public class ExperimentalFarm
{
    private Plot[][] farmPlots;

    public ExperimentalFarm(Plot[][] p)
    {
        /* implementation not shown */
    }

    /** Returns the plot with the highest yield for a given
        crop type, as described in part (a). */
    public Plot getHighestYield(String c)
    {
        /* to be implemented in part (a) */
    }

    /** Returns true if all plots in a given column in the
     *  two-dimensional array farmPlots contain the same
     *  type of crop, or false otherwise, as described in
     *  part (b) */
    public boolean sameCrop(int col)
    {
        /* to be implemented in part (b) */
    }
}
```

Write the `getHighestYield` method, which returns the `Plot` object with the highest yield among the plots in `farmPlots` with the crop type specified by the parameter `c`. If more than one plot has the highest yield, any of these plots may be returned. If no plot exists containing the specified type of crop, the method returns `null`.

Assume that the `ExperimentalFarm` object `f` has been created such that its `farmPlots` array contains the following `cropType` and `cropYield` values.

	0	1	2
0	"corn" 20	"corn" 30	"peas" 10
1	"peas" 30	"corn" 40	"corn" 62
2	"wheat" 10	"corn" 50	"rice" 30
3	"corn" 55	"corn" 30	"peas" 30

The following are some examples of the behavior of the `getHighestYield` method.

Method Call	Return Value
`f.getHighestYield("corn")`	`farmPlots[1][2]`
`f.getHighestYield("peas")`	`farmPlots[1][0]` ***or*** `farmPlots[3][2]`
`f.getHighestYield("bananas")`	`null`

(a) Write the `getHighestYield` method below.

```
/** Returns the plot with the highest yield for a given crop type, as
described in part (a). */
    public Plot getHighestYield(String c)
    {
        Plot maxP = null;
        int maxY = -1;

        for (int row = 0; row < farmPlots.length; row++)
        {
            for (int col = 0; col < farmPlots[0].length;
                 col++)
            {
                Plot p = farmPlots[row][col];
                if (p.getCropType().equals(c) &&
                    p.getCropYield() > maxY)
                {
                    maxP = p;
                    maxY = p.getCropYield();
                }
            }
        }

        return maxP;
    }
```

Points earned:
+1 [Skill 3.E] Traverses all elements of the farmPlots array (without bounds errors)
+1 [Skill 3.E] Accesses an element of farmPlots in the context of a loop
+1 [Skill 3.A] Accesses the crop type and crop yield of an element of farmPlots
+1 [Skill 3.C] Compares the crop type of an element to the parameter c
+1 [Skill 3.E] Implements find-max algorithm in the context of this 2D array:
Initializes, compares, and updates maximum yield and Plot containing maximum yield
+1 [Skill 3.B] Returns Plot containing max yield (or null, if appropriate)

General Penalties:
-1 (v) Array/collection access confusion ([] get)
-1 (w) Extraneous code that causes side-effect
(e.g., printing to output, incorrect precondition check)
-1 (x) Local variables used but none declared
-1 (y) Destruction of persistent data
(e.g., changing value referenced by parameter)

General penalty (y) is incurred if farmPlots is modified

Write the `sameCrop` method, which returns `true` if all the plots in a given column of `farmPlots` grow the same crop and returns `false` otherwise.

The following are two examples of the behavior of `sameCrop`.

- The method call `f.sameCrop(0)` returns `false` because the values of `cropType` for the elements of column `0` (`"corn"`, `"peas"`, `"wheat"`, and `"corn"`) are not all the same.
- The method call `f.sameCrop(1)` returns `true` because the values of `cropType` for all elements of column `1` are the same (`"corn"`).

(b) Write the `sameCrop` method below.

```
/** Returns true if all plots in a given column in the two-dimensional
 *  array farmPlots contain the same type of crop, or false otherwise,
 *  as described in part (b).*/

  public boolean sameCrop(int col)
  {
     String c = farmPlots[0][col].getCropType();
     for (int row = 1; row < farmPlots.length; row++)
     {
        if (!farmPlots[row][col].getCropType()
            .equals(c))
        {
           return false;
        }
     }
     return true;
  }
```

Points earned:
+1 [Skill 3.E] Traverses the given column of farmPlots (without bounds errors)
+1 [Skill 3.A] Accesses the crop types of two elements of farmPlots
+1 [Skill 3.C] Determines value to be returned as false if any two crop types are different and true if all are the same

General Penalties:
-1 (v) Array/collection access confusion ([] get)
-1 (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)
-1 (x) Local variables used but none declared
-1 (y) Destruction of persistent data (e.g., changing value referenced by parameter)