PPA Assignment 10

Wonjoon Seol, Computer Science with Intelligent Systems, K1631098

December 19, 2016

1 Introduction

In this assignment we simulate an agriculture activity. A farmer can plant any type of crops, and harvest them using a number of harvesters to earn profit. I need to demonstrate my ability to use inheritance and override necessary methods, as well as everything we have learnt in this semester. (Topics from week 1,. . . ,5 and 7,. . . ,11).

2 Pseudocode

2.1 Class Crop

Pseudocode 1: This class represents type and value of crops.

```
Initialise private String type
Initialise private integer value
Define Part
    Set row
    Set column
Define Crop
    Set String type
    Set integer value
Define getValue
    return value
```

2.2 Class Field

Pseudocode 2: This class represents Field where crops will be planted.

```
Initialise public static final integer $MAX\_NUM\_CROPS$ and set it 10
Initialise private ArrayList<Crop> crops
Define Field
    Initialise ArrayList crops type Crop
    plant crops with supplied name and value
Define plant
    Initialise boolean $flag$ to be false
    if ArrayList crops is empty then
        for integer i between 0 and $MAX\_NUM\_CROPS$ - 1 do
            Initialise Crop with supplied type and value and add it to ArrayList crops
            Set $flag$ true
        end
    end
    Return $flag$
Define harvest
    Initialise integer $profit$ to be false
    if ArrayList crops is not empty then
        for each crop in ArrayList crops do
            Add value of crop to $profit$
        end
    end
    empty ArrayLIst crops
    Return $profit$
```

2.3 Class Harvester

Pseudocode 3: This class represents characteristics of a harvester.

```
Initialise private integer $fuelTankSize$
Initialise private integer <Crop> topSpeed
Define Harvester
    Set $fuelTankSize$
    Set topSpeed
Define calculateHarvestingCapacity
    Return $fuelTankSize$ + topSpeed
Define getTopSpeed
    Return topSpeed
Define getFuelTankSize
    Return $getFuelTankSize$
```

2.4 Class CombineHarvester

Pseudocode 4: This class represents CombineHarvester, type Harvester.

```
Initialise private integer length
Define CombineHarvester
    Call superclass Harvester constructor with supplied fuelTankSize and topSpeed
    Set length
Define calculateHarvestingCapacity
    Return $(topSpeed + fuelTankSize) \times length$
```

2.5 Class Farm

Pseudocode 5: This class represents a farm with multiple fields and harvesters.

```
Initialise private ArrayList<Field> fields
Initialise private ArrayList<Harvester> harvesters
Initialise integer profit
Define Farm
    Initialise ArrayList crops type Crop
    Initialise ArrayList crops type Crop
Define addField
    Initialise new Field with supplied type and value add to ArrayList fields
Define addHarvester
    Add a supplied harvester to ArrayList harvesters
Define getProfit
    Return profit
Define harvest
    Initialise integer totalCapacity to be 0
    for integer i between 0 and size of ArrayList harvesters - 1 do
        calculate harvesting capacity of all harvesters
        Add total harvesting capacity to totalCapacity
        if totalCapacity is less than or equal to size of ArrayList fields then
            for integer i between 0 and totalCapacity - 1 do
                Add value returned from harvest ith element in fields to profit
            end
        else
            for integer i between 0 and size of fields - 1 do
                Add value returned from harvest ith element in fields to profit
            end
        end
    end
```

2.6 Class Harvest

Pseudocode 6: This class is going to drive our program.

```
Initialise new Farm $farm$
Initialise new Harvester and add it to $farm$
Initialise new CombineHarvester and add it to $farm$
for integer i between 0 and 4 do
    Add field with Corn, each value of 20 to farm
    Add field with Barley, each value of 20 to farm
    Add field with Wheat, each value of 20 to farm
    Add field with Oat, each value of 20 to farm
end
harvest $farm$
Print profit of $farm$
```

3 Class Diagram

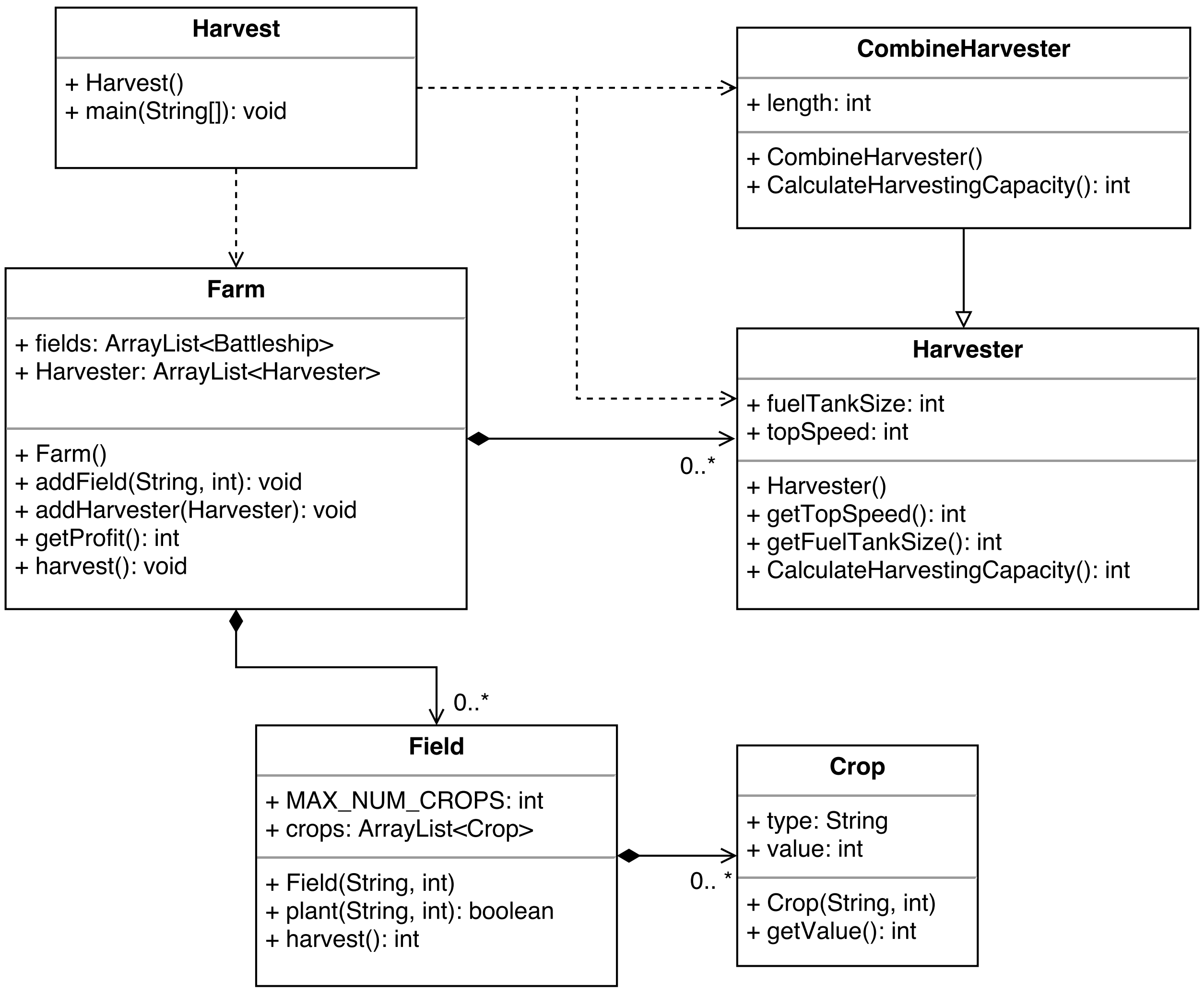

Figure 1: Harvest Class Diagram.

4 Description

The class Crop represent basic type and value of a crop, which the class Field stores them inside one of its field. No other instance of this class is shared across other classes, so the class Field has a composite relationship to the Class Crop.

The Class Field has static final field MAX_NUM_CROPS as every field in this assignment has same number of crops. The *plant* method checks whether there is a crop already in the field and return false if planting a new crop is not successful. Likewise, the method *harvest* checks whether the crop is not already sold and then add each value of crop to calculate its profit.

The Class CombineHarvester is inherited from the class Harvester. The $calculateHarvestingCapacity$ is overridden because CombineHarvester has greater harvesting capacity.
Initially, I used *setHarvestingCapacity* method to set the harvesting capacity. However, this allows someone else to change the harvesting capacity outside of the class. Therefore, the method was removed and the field *harvestingCapacity* is now encapsulated by these two classes.

The Class Farm can add instances of Field and Harvester to its ArrayList. The most important method is *harvest*. This first checks for total capacity of all harvesters and then check whether this size is greater than the size of $fields$. If it is less than the size of $fields$ then the farmer only harvest and sell the crops within the possible fields.

Our driver Class Harvest add harvesters and determine the types of crops to be added to the fields. Finally, the farmer harvest his farm and prints its profit.