# Inheritance and Modeling

This recitation has two distinct parts. In the first part you will gain experience with inheritance in Java using an example very similar to the example discussed in class. In the second part of this recitation you will practice object-oriented analysis and design, using UML to create a domain model, an interaction diagram, and an object model from a textual description of a problem domain.

## Inheritance

Examine the classes in the `edu.cmu.cs.cs214.rec03.inheritance` package. The following code is from `FarmMain.java`.

1. Why doesn't the following line compile?

```
Animal animal = new Animal();
```

2. Even though sheep is a `Sheep` and cow is a `Cow`, how can we be sure that we can call `makeProduct()` on both objects? See `FarmMain#makeProduct(Animal)`.

```
makeProduct(sheep);
makeProduct(cow);
```

3. What will the following print?

```
animalSheep.makeNoise();
animalCow.makeNoise();
```

4. Which of the following assignments will fail?

```
Sheep newSheep0 = (Sheep) animalSheep;
Sheep newSheep1 = (Sheep) sheep;
Cow newCow0 = (Cow) animalSheep;
```

5. How many legs does each animal have? (What is the output of the following?)

```
cow.countLegs();
sheep.countLegs();
```

6. What happens when the following code is run? Why?

```
Animal[] animals = new Animal[NUM_ANIMALS];
Animal[] sheeps = new Sheep[NUM_ANIMALS];

for (int i = 0; i < NUM_ANIMALS; i++) {
    if (i % 2 == 1) {
        animals[i] = new Cow();
    } else {
        animals[i] = new Sheep();
    }
    sheeps[i] = new Cow();
}
```

7. While iterating through animals, what `makeNoise()` methods are called?

```
for (int i = 0; i < NUM_ANIMALS; i++) {
    animals[i].makeNoise();
}
```

8. What will happen when the following code is run?

```
animalSheep.action();
animalCow.action();
```

9. Though `MyFarm` doesn't contain the method `addAnimal()`, there is no error in the following lines (`oldMacDonald` is an instance of `MyFarm`). Why not?

```
oldMacDonald.addAnimal(animalSheep);
oldMacDonald.addAnimal(sheep);
```

## Object-oriented analysis and design

In this section, you will analyze a problem domain, using UML to produce a domain model, an interaction diagram, and an object model from a textual description of a problem domain. Specifically, your goal is to design the core of a system to track inventory within a warehouse:

> The warehouse contains items, boxes, and workers. Each item has an immutable (unchangeable) name, weight, and dimensions (length, width, and height), as well as a mutable location inside the warehouse. Each box has a weight (the weight of the empty box), dimensions, location, and a weight limit. A box can hold arbitrary other items (including other boxes) as long as the items can be packed within the box's dimensions and the total weight of the items does not exceed the box's weight limit. A worker also has a location and a weight limit. A worker can carry only one item at a time, but that item can be a box containing other items. Workers cannot be placed in boxes.

1. Based on the description above, use a UML class diagram to create a domain model for the inventory tracking system. Remember: a domain model represents real-world concepts in the so-called problem space, not software concepts in the solution space of the design.

2. Suppose you had a laptop weighing 3 pounds and an extension cord weighing 0.5 pounds in the warehouse. The laptop is in a small box that weighs 1 pound. The small box and the extension cord are in a large box weighing 10 pounds. The system now wants to determine whether a given worker is allowed to lift the large box.

   Using a UML communication diagram or a UML sequence diagram, create an interaction diagram for the scenario above that clarifies the dynamic behavior of the system when it ensures that a worker does not lift too heavy a box.

3. Use a UML class diagram to create an object model for the inventory tracking system. Specifically, your design should model the concepts above, and given a collection of items to be shipped, enable the warehouse to instruct a worker how to move items around the warehouse to pack the items into a box or boxes. (For example, your model should enable optimization algorithms to be implemented later, although you do not need to actually write any optimization algorithms here).