# Interacting with the Superclass

You should complete this programming project independently. You may use your classroom notes, Tricky Code Cheat Sheet, 4 Commandments of Scope, or any other guides you’ve written. You may also reference your textbook, posters and sample code posted around the room, or guides on the internet. All code must be your own.

To receive full credit on this project, you must write pseudocode or structure diagrams to document your planning process. Remember to include comments in your code to explain code to your instructor.

## Exercise 1

In the space below, record your answer to the following questions. You may provide examples of code if it helps you answer the question.

1. What is the difference between overloading and overriding a method?
2. When should you use the super keyword?
3. What is code reuse? How does inheritance help achieve code reuse?

Exercise 2

To answer the questions in Exercise 2, consider the following class:

//Represents a university student.

public class Student{

private String name;

private int age;

public Student(String n, int a){

name = n;

age = a;

}

public void setAge(int ag){

age = ag;

}

}

Also consider the following partial implementation of a subclass of *Student* to represent undergraduate students at a university:

public class UndergraduateStudent extends Student{

private int year;

…

}

1. Can the code in the *UndergraduateStudent* class access the *name* and *age* fields it inherits from *Student*? Can it call the *setAge* method? Why or why not?
2. Write a constructor for the *UndergraduateStudent* class that accepts a name as a parameter and initializes the *UndergraduateStudent*’s state with that name, an *age* value of 18, and a *year* value of 0.
3. Write a version of the *setAge* method in the *UndergraduateStudent* class that not only sets the *age*, but also sets the *year* field to *age* -18.
4. Write an *equals* method that tests whether an *UndergraduateStudent* and *Student* are the same age.

Exercise 3

To complete Exercise 3, you should reference the *Drink* superclass. If you do not have the *Drink* superclass documented in your notes, you should ask for a copy of this class from your instructor.

1. Write the class *SugaryDrink* to accompany the *Drink* superclass. *SugaryDrink* can be categorized as containing juice or not. *SugaryDrink* has an additional method called *printDrinkLabel* that prints “Contains no actual juice.” if the drink doesn’t contain juice, or “Contains real fruit juice!” if the drink contains juice. Be sure to interact with the superclass wherever it is appropriate.
2. Write a class *SugarFreeDrink* to accompany the other beverage classes. *SugarFreeDrink* drinks contain no sugar, may or may not contain artificial sweetener, and are larger in serving size than other drinks, coming in 20 oz. servings. *SugarFreeDrink* has an additional method called *printDrinkLabel* that prints “This drink is not all natural.” if the drink contains artificial sweeteners. If no artificial sweeteners are present, *printDrinkLabel* outputs “This drink contains no artificial sweeteners.”
3. Write a class *SportsDrink* to accompany the other beverage classes. *SportsDrink* drinks are like other sugared drinks as they contain some type of sugar for energy. *SportsDrink* also contain electrolytes to replenish what is lost during sweating. Sports drinks have a serving size of 12 ounces even though many come in varying sizes.
4. Write an *equals* method that lets coaches know if the serving size of different instances of the *SportsDrink* class have the same number of ounces.