<u>csTask50 Assignment</u>

The activities in this assignment require you to create and modify Java programs using BlueJ. The BlueJ project csTask50 already contains various classes for you to work on.

Follow the directions given for each part. At the end of class, all modified programs (class files) must be in your project, and the project must be copied to your shared Dropbox folder.

When questions are posed, write your responses as full sentences in the Readme.txt file in your project.

There are two parts to this assignment:

I. Overriding the `equals` Method
II. Test Questions

I. Overriding the `equals` Method

The instance variables of the Player class hold information about an athlete: name, team, and uniform number. This class also provides a method that prompts for and reads in this information from the keyboard. ComparePlayers is a skeleton of a program that uses the Player class to read in information about two baseball players and determine whether or not they are the same player.

1. Fill in the missing code in ComparePlayers so that it reads in two players and prints "Same player" if they are the same, "Different players" if they are different. Use the *equals* method, which Player inherits from the Object class, to determine whether two players are the same. Are the results what you expect?

2. The problem above is that as defined in the Object class, *equals* does an address comparison. It says that two objects are the same if they live at the same memory location, that is, if the variables that hold references to them are aliases. The two Player objects in this program are not aliases, so even if they contain exactly the same information they will be "not equal." To make *equals* compare the actual information in the object, you can override it with a definition specific to the class. It might make sense to say that two players are "equal" (the same player) if they are on the same team and have the same uniform number.

a) Use this strategy to define an `equals` method for the Player class. Your method should take a Player object and return true if it is equal to the current object, false otherwise.

b) Test your ComparePlayers program using your modified Player class. It should give the results you would expect.

II. Test Questions

In this exercise you will use inheritance to read, store, and print questions for a test. You will find it helpful to read this entire exercise description before beginning to plan and code.

First, write an abstract class TestQuestion that contains the following:

a) A `protected` String variable that holds the test question. Recall that the `protected` modifier provides public visibility for a variable or method only within the package.
b) An abstract method `public abstract void readQuestion(Scanner fileScan)` to read the question. Note that this method takes a Scanner variable as a parameter. This is explained further below. Since an abstract method is only a header, you don't have to worry about the rest of the implementation of this method yet.

Now define two subclasses of TestQuestion: Essay and MultChoice. Essay will need an instance variable to store the number of blank lines needed after the question (answering space). MultChoice will not need this variable, but it will need an array of Strings to hold the choices along with the main question.

Assume that the input for the program is in an input file as follows, with each item on its own line:

- type of question (character, m=multiple choice, e=essay)
- number of blank lines for essay, number of answer choices for multiple choice (integer)
- choice 1 (multiple choice only)
- choice 2 (multiple choice only)
- … etc., (multiple choice only) more choices as needed

The very first item of input, before any questions, is an integer indicating how many questions will be entered. For example, the input on the following page represents five questions: an essay question requiring five blank lines, a multiple choice question with four choices, then two more essay questions requiring five blank lines and three blank lines, respectively, then a multiple choice with three choices.

You will need to write `readQuestion` methods for the MultChoice and Essay classes that read information in this format. Since the type of question needs to be determined before the MultChoice or Essay constructor is run, the character that identifies what kind of question it is will be read by the driver class. As such, the Scanner variable that reads the lines of data from the file can be set up in the driver and passed to the `readQuestion` method as a parameter. Because the input file has a mixture of data types, it is recommended that you use the nextLine() method of the Scanner class throughout the program.

You will also need to write `toString` methods for the MultChoice and Essay classes that return nicely formatted versions of the questions. For example, the answer choices for a MultChoice question should be lined up, labeled a), b), c), …etc, and indented in MultChoice. You can assume that there will be no more than six choices for each MultChoice. If there are fewer than six, only print those that apply (e.g., if there are two choices, only print an a) and b) choice.)

The driver class WriteTest should create an array of TestQuestion objects. It should read the questions from the input file, first reading an integer that indicates how many questions are coming. It should create a MultChoice object for each multiple choice question and an Essay object for each essay question and store each object in the array. (Since it's an array of TestQuestion and both Essay and MultChoice are subclasses of TestQuestion, objects of both types can be stored in the array.) When all of the data has been read, it should use a loop to print the questions, numbered, in order. A shell of this class is provided in the BlueJ project. Note that the Scanner variable used to read from the file is already set up.

Use the data in `testdata.txt` (shown on the following page and included in the BlueJ project) to run your program. Some smaller input files (test2, test3, test4) are in the project to use as you test the various parts of your program before running it on the entire file.

testdata.txt

```
5
e
5
Why does the constructor of a subclass have to call the constructor of its parent class?
m
4
Which of the following is not a legal identifier in Java?
guess2
2ndGuess
_guess2_
Guess
e
5
What does the “final” modifier do?
e
3
Java does not support multiple inheritance. This means that a class cannot do what?
m
3
A JPanel has an addMouseListener method because JPanel is a subclass of:
JComponent
JApplet
Object
```