CS-141 Introduction to Programming
Creating an Inheritance Hierarchy

Goals: Practice writing and using inheritance.

In this lab, you are given a `Card` class (not a playing card!) as a superclass. Based on this class, you will implement a hierarchy of related classes for ID card, driver's license and a calling card. You will then create a Billfold class in which to hold your cards.

Instructions

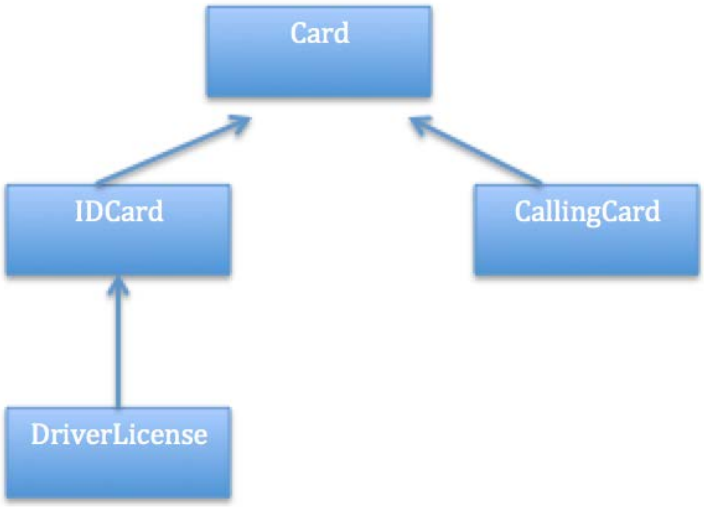
Step 1: Create a new Netbeans project, e.g. called BillFoldProject. When you create the project, be sure to check the "Create main class" box so that Netbeans creates a class containing the main method.

Step 2: Add a new class called `Card` and copy the code below into the class (or see link on assignment page).

```
public class Card {

    private String name;

    public Card() {
        name = "";
    }

    public Card(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public boolean isExpired() {
        return false;
    }

    public String toString() {
        return "name: " + name;
    }
}
```

Step 3: Test your code by creating a `Card` object in `main()` and calling the various methods.

Step 4: The class inheritance relationship is shown in the following picture:



where the unique feature for each card is listed in the following table.

Class	Data
`IDCard`	ID number (`String`)
`CallingCard`	Card Number (`String`), PIN(`int`)
`DriverLicense`	Expiration year (`int`)

Write class declarations for each of these subclasses. For each subclass, you will need to add protected (or private) instance variables, a constructor, getters&setters, and toString() methods. The constructors and toString() methods will build on the corresponding methods in the superclass by using the keyword `super`.

Step 5: Test each of your subclasses by creating objects of each type in `main()` and calling the various methods. The `toString` methods in each subclass should be written by building on the `toString` method of the super class. For example, for the following code in `main()`:

```
IDCard card1 = new IDCard("Mary Kay", "1234567");
System.out.println(card1);
CallingCard card2 = new CallingCard("Omega Card", "301233985945", 1030);
System.out.println(card2);
DriverLicense card3 = new DriverLicense("John Doe", "08-097654", 2007);
System.out.println(card3);
```

Your output should look like:

```
ID[name: Mary Kay, id:1234567]
Calling Card[name: Omega Card, Card #:301233985945, pin: 1030]
Driver's License[ ID[name: John Doe, id:08-097654], expYear: 2007]
```

Step 6: Suppose you have a billfold with 3 slots for cards. The slots may or may not be empty. Create a class, called `Billfold`, to represent your billfold. In `Billfold`, the slots are represented by an instance member variable of type `Card` array with size 3. Because the array can contain objects of type Card, you can store objects of type Card along with objects which are subclasses of Card. This is the concept of ***polymorphism***. You need to add your array to the `Billfold` class as a member instance variable. If a slot is empty, it will be set to null. Initially, all the slots are empty. Add the following methods:

- `addCard(Card c)` which finds the first empty slot, and places the card there. If there are no empty slots, nothing happens.
- `toString()` which prints out all of the information about all of the cards in the Billfold, one card per line. If a slot is empty, the word "empty" should be printed. See Step 7 for example.

Step 7: Delete the previous test code you had in `main()` and paste in the code below. (this is also available in a link from the assignment page). If you have implemented the code correctly, the `Actual` output and `Expected` output should be identical when you run the program:

```
public static void main(String[] args) {
        // Create an empty billfold
        Billfold b = new Billfold();
        System.out.println("**Actual: ");
        System.out.println(b);
        System.out.println("**Expected: ");
        System.out.println("Billfold:\nempty\nempty\nempty\n");

        // Create two Card objects and add them to the billfold
        IDCard card1 = new IDCard("Mary Kay", "1234567");
        CallingCard card2 = new CallingCard("Omega Card", "301233985945", 1030);
        b.addCard(card1);
        b.addCard(card2);
        System.out.println("**Actual: ");
        System.out.println(b);
        System.out.println("**Expected: ");
        System.out.println("Billfold:\nID[name: Mary Kay, id:1234567]\n"
                + "Calling Card[name: Omega Card, Card #:301233985945, pin: 1030]\n"
                + "empty\n");

        // create another card and add to the billfold
        DriverLicense card3 = new DriverLicense("John Doe", "08-097654", 2007);
        b.addCard(card3);
        System.out.println("**Actual: ");
        System.out.println(b);
        System.out.println("**Expected: ");
        System.out.println("Billfold:\n"
                + "ID[name: Mary Kay, id:1234567]\n"
                + "Calling Card[name: Omega Card, Card #:301233985945, pin: 1030]\n"
                + "Driver's License[ ID[name: John Doe, id:08-097654], expYear: 2007]\n");
    }
```

Step 8: Demo your code using the above `main()` and submit via WISE following the directions given there. Note, you will need to zip together the entire Netbeans project and add it as an attachment. If you don't know how to do this, please ask the lab assistant or instructor.