

Inheritance

1. Consider using the following `Card` class.

```
public class Card {
    public Card(){ name = "";}
    public Card(String n){ name = n;}
    public String getName(){ return name;}
    public boolean isExpired(){ return false;}
    public String format(){ return "Card holder: " + name;}
    private String name;
}
```

Use this class as a superclass to implement a hierarchy of related classes:

Class	Data
IDCard	ID number
Calling Card	Card number, PIN
Driver License	Expiration year

Write definitions for each of the subclasses. For each subclass, supply private instance variables. Leave the bodies of the constructors blank for now.

Calling the Superclass Constructor

2. Implement constructors for each of the three subclasses. Each constructor should call the superclass constructor to set the name. Here is one example:

```
public IDCard(String n, String id)
{
    super(n);
    idNumber = id;
}
```

Overriding Methods

3. Supply the implementation of the `format` method for the three subclasses. The methods should produce a formatted description of the card details. The subclass methods should call the superclass `format` method to get the formatted name of the cardholder.

4. Devise another class, `Billfold`, which contains slots for two cards, `card1` and `card2`, a method `void addCard(Card)` and a method `String formatCards()`.

The `addCard` method checks if `card1` is `null`. If so, it sets `card1` to the new card. If not, it checks `card2`. If both cards are set already, the method has no effect.

Of course, `formatCards` invokes the `format` method on each non-`null` card and concatenates the resulting strings.

What is your `Billfold` class?

5. Write a class with a test program that adds two objects of different subclasses of `Card` to a `Billfold` object. Print the results of the `formatCards` methods.

What is the code for your test program?

6. What is the output of your test program?

7. Explain why the output of your program demonstrates polymorphism.

8. The `Card` superclass defines a method `isExpired`, which always returns `false`. This method is not appropriate for the driver license. Supply a method `DriverLicense.isExpired()` that checks if the driver license is already expired (i.e., the expiration year is less than the current year).

To find out the current year, you can use the `get` method of the class `Calendar`. For example, if you create a `Calendar` as follows:

```
GregorianCalendar calendar = new GregorianCalendar();
```

Then, you can obtain the current year using

```
calendar.get(Calendar.YEAR)
```

What is the code for your `isExpired` method?

9. The ID card and the phone card don't expire. What should you do to reflect this fact in your implementation?
10. Add a method `getExpiredCardCount`, which counts the number of expired cards in the billfold, to the `Billfold` class.
11. Write a test class that populates a billfold with a phone card and an expired driver license. Then call the `getExpiredCardCount` method. Run your test program to verify that your method works correctly.

What is your test program?

The `toString` method

12. Define `toString` methods for the `Card` class and its three subclasses. The methods should print:
 - the name of the class
 - the values of all instance fields (including inherited fields)

Typical formats are:

```
Card[name=Edsger W. Dijkstra]
CallingCard[name=Bjarne Stroustrup][number=4156646425,pin=2234]
```

Give the code for your `toString` methods.

The `equals` method

13. Define `equals` methods for the `Card` class and its three subclasses. Cards are the same if the objects belong to the same class, and if the names and other information (such as the expiration year for driver licenses) match.

Give the code for your `equals` methods.