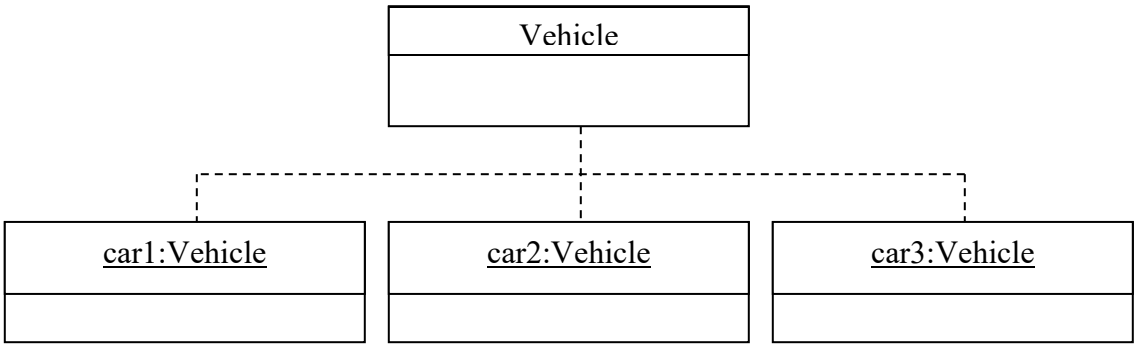
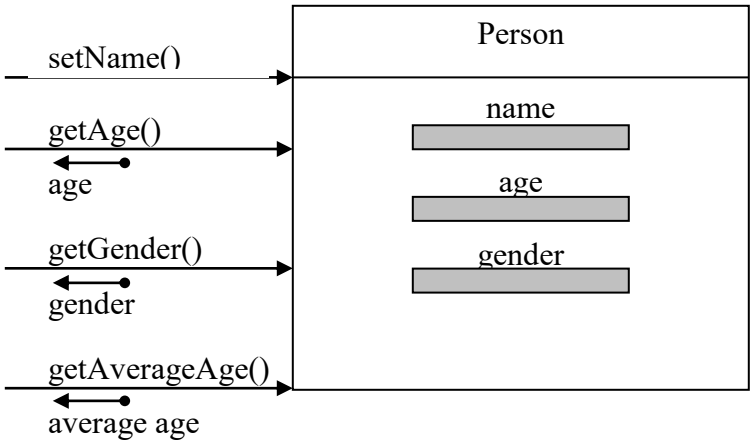
1.1 Graphically represent a Vehicle class and three Vehicle objects named car1, car2, and car3.



1.2 Graphically represent a Person class with the following components:

- Instance variables name, age, and gender.
- Instance methods setName, getName, and getAge.
- Class method getAverageAge.



1.3 Design a CD class where a CD object represents a single music CD. What kinds of information (artist, genre, total playing time, etc.) do you want to know about a CD? Among the information in which you are interested, which are instance variables? Are there any class variables or class constants?

State

Album Name	*instance variable*
Artist	*instance variable*
Genre	*instance variable*
Total Playing Time	*instance variable*
Maximum Playing Time	*class constant*

Behaviors

getAlbumName(), setAlbumName()	*instance methods*
getArtist(), setArtist()	*instance methods*
getGenre(), setGenre()	*instance methods*
getPlayingTime(), setPlayingTime()	*instance methods*
getMaxPlayingTime()	*class method*

Students may also include a list of songs on the CD and methods to access them. A song itself would probably have its own class to represent it. However this concept is more advanced than is necessary at this stage.

1.4 Suppose the Vehicle class in Exercise 1 is used in a program that keeps track of vehicle registration for the Department of Motor Vehicles. What kinds of instance variables would you define for such Vehicle objects? Can you think of any useful class variables for the Vehicle class?

Instance variables

owner
licenseID
registrationNumber
make
model
color
value

Class variables

At this stage, the number of total vehicles could be thought to belong to the class. Information relating to the length of a licenseID or registrationNumber could be stored as class constants. Aside from these, there are no obviously necessary traits for which all vehicles share the same value or which the class may need to function.

1.5 Suppose the following formulas are used to compute the annual vehicle registration fee for the vehicle registration program in Exercise 1.4:

- For cars, the annual fee is 2 percent of the value of the car.
- For trucks, the annual fee is 5 percent of the loading capacity (in pounds) of the truck.

Define two new classes Car and Truck as subclasses of Vehicle.
Hint: Associate class and instance variables common to both Car and Truck to Vehicle.

Vehicle Class

See Exercise 1.4

Car Class (subclass of Vehicle)

registrationRate	*class constant*

Note: Value is already an instance variable in Vehicle since all vehicles have some value.

Truck Class (subclass of Vehicle)

registrationRate	*class constant*
loadingCapacity	*instance variable*

1.6 Consider a student registration program used by the registrar's office. The program keeps track of students who are registered for a given semester. For each student registered, the program maintains the student's name, address, and phone number; the number of classes in which the student is enrolled and the student's total credit hours. The program also keeps track of the total number of registered students. Define instance and class variables of a Student class that is suitable for this program.

Instance variables

name
address
phoneNumber
numClassesThisSemester
totalCreditHours

Class variables

numStudentsRegistered

1.7 Suppose the minimum and maximum number of courses for which a student can register are different depending on whether the student is a graduate, undergraduate, or work-study student. Redo Exercise 6 by defining classes for different types of students. Relate the classes using inheritance.

Student

See Exercise 1.6

GraduateStudent (subclass of Student)

maximumHours	*class constant*
minimumHours	*class constant*

UndergraduateStudent (subclass of Student)

maximumHours	*class constant*
minimumHours	*class constant*

WorkStudyStudent (subclass of Student)

maximumHours	*class constant*
minimumHours	*class constant*

1.8 Imagine you are given a task of designing an airline reservation system that keeps track of flights for a commuter airline. List the classes you think would be necessary for designing such a system. Describe the data values and methods you would associate with each class you identify. *Note:* For this exercise and Exercises 9 through 12, we are not expecting you to design the system in complete detail. The objective of these exercises is to give you a taste of thinking about a program at a very high level. Try to identify about a half dozen or so classes, and for each class, describe several methods and data members.

Database

Data Members
Collection of clients
Collection of flights

Methods
Accessors (get___()) and Mutators (set___()) for clients and flights[1]
Make reservation
Add new flight

Client

Data Members
Name
Address
Phone
Collection of reservations
BillingInformation

Methods
Accessors and Mutators for name, address, BillingInformation, and collection of reservations
Add reservation

Flight

Data Members
Departure city
Arrival city
Departure time
Arrival time
Seats available
Aircraft type

---

[1] Accessors and Mutators (also called gets and sets) allow other people to use your classes data members while allowing you to control just how they access them. This allows you to perform various activities like bounds checking (making sure the value set is not illegal, such as –6 for an age data member). This is part of the concept of encapsulation and is fundamental to the object-oriented paradigm.

Collection of passengers

Methods
Accessors and Mutators for departure and arrival information, seats available, aircraft type
Compute flight time
Compute ticket price (may vary with departure date, time, seats available…)

Reservation

Data Members
Client
Flight
Paid Status (i.e., true/false)

Methods
Accessors and Mutators for client and flight

BillingInformation

Data Members
Name
Billing Address
Credit Card number
Credit Card expiration
Credit Card type

Methods
Accessors and Mutators for all data members

Note: This is a high-level design and by no means the only appropriate one. When designed in more detail, there will be ways to eliminate the duplicate data.

1.9 Repeat Exercise 8, designing a university course scheduling system. The system keeps track of classes offered in a given quarter, the number of sections offered, and the number of students enrolled in each section.

Quarterly Database

Data Members
Collection of courses

Methods
Add Course
Remove Course

Course

Data Members
Collection of Sections
Title
Code
College
Department

Methods
Add Section
Remove Section
Accessors and Mutators for all data members

Section

Data Members
Maximum number of students
Current number of students
Section ID

Methods
Add Student
Remove Student
Accessors and Mutators for all data members

1.10 Repeat Exercise 8, designing the state Department of Motor Vehicles registration system. The system keeps track of all licensed vehicles and drivers. How would you design objects representing different types of vehicles (e.g., motorcycles and trucks) and drivers (e.g., class A for commercial licenses and class B for towing vehicles)?

Vehicle

Data Members
VIN
Class type (A, B, etc.)
Make
Model
Year
Registration fee
Owner

Methods
Accessors and Mutators for all data members

Note: Could also implement with several subclasses of Vehicle, one for each class (car, commercial truck, livery, etc.) especially if each type required some specialized information.

Driver

Data Members
License number
Name
Address
Collection of Vehicles
Collection of Violations

Methods
Add Vehicle/Violation
Remove Vehicle/Violation
Accessors and Mutators for all data members

Violation

Data Members
Number
Date
Location
Charge
Officer

Methods
Accessors and Mutators for all data members

1.11 Repeat Exercise 8, designing a sales tracking system for a fast-food restaurant. The system keeps track of all menu items offered by the restaurant and the number of daily sales per menu item.

Menu

Data Members
Collection of menu items
Total Sales

Methods
Get Menu Item
Add Menu Item
Delete Menu Item

MenuItem

Data Members
Name
Price
Total Sales

Methods
Accessors and Mutators for name, price, and total sales

1.12 When you write a term paper, you have to consult many references: books, journal articles, newspaper articles, and so forth. Repeat Exercise 8, designing a bibliography organizer that keeps track of all references you used in writing a term paper.

Document

Data Members
Collection of sources

Methods
Add Source
Delete Source

Source

Data Members
Author
Title
Publisher
Copyright Date
City of Publication
Volume (or Issue)
Pages Used

Methods
Print Bibliography Entry

It would be common practice for there to be several different subclasses of Source, each with a slightly different implementation of Print Bibliography Entry, for example: paper in a conference proceedings, article in a journal, journal, book, article in a magazine, magazine, etc.

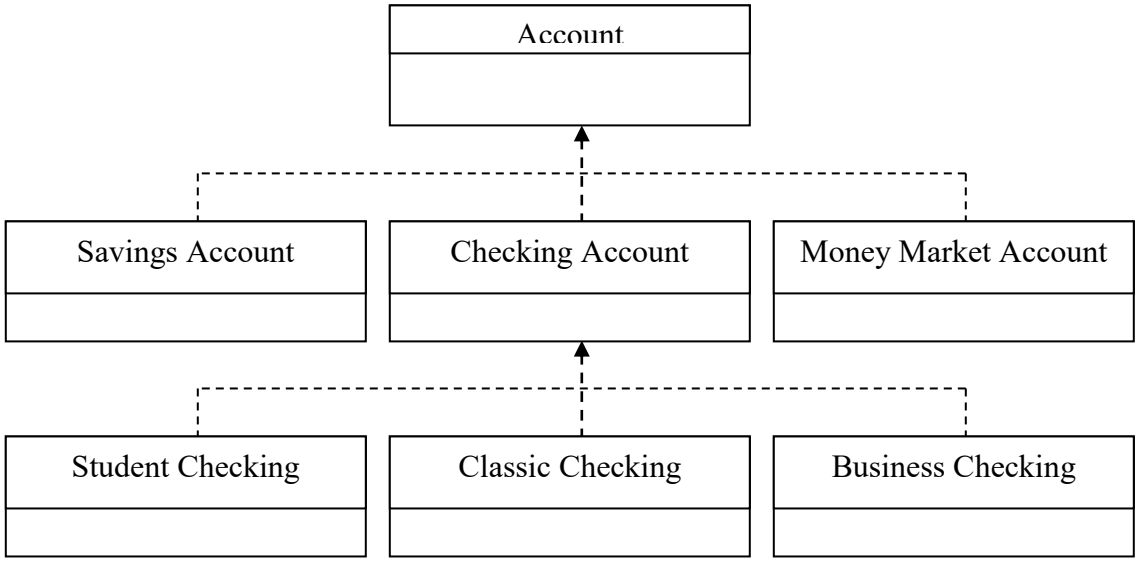
1.13 Consider the inheritance hierarchy given in Figure 1.12. List the features common to all classes and the features unique to individual classes. Propose a new inheritance hierarchy based on the types of accounts your bank offers.

Possible Common Features:

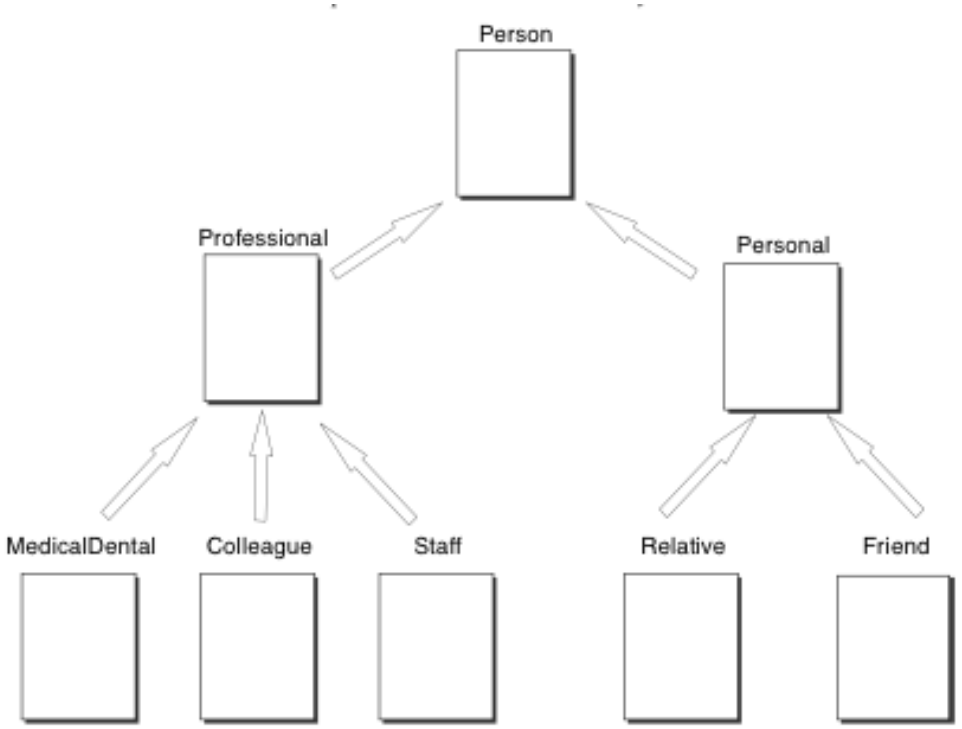
Account Number
Opening Balance
Opening Date
Current Balance
Minimum Balance
Owner(s)

Possible Unique Features:

Number of free withdrawals
Fee per withdrawal
Interest Rate
Check writing privileges
Debit Card information



1.14 Consider a program that maintains an address book. Design an inheritance hierarchy for the classes such as Person, ProfessionalContact, Friend, and Student that can be used in implementing such a program.



1.15 Do you think the design phase is more important than the coding phase? Why or why not?

The design phase is much more important than the coding phase. Design has a huge impact on not only the ease and speed of implementation during the coding phase but also on the testing and maintenance phases. A proper design allows testing to be done in terms of structural components, which makes bugs easier to trace back to the improper implementation. Furthermore, a good design is far more maintainable. Maintenance is considered the most expensive of all the phases, thus a more easily maintained project is considerably less expensive. It is important to remember that maintenance includes more than just fixing bugs; extension of the original design to meet new requirements is a common and important task.

1.16 How does the quality of design affect the total cost of developing and maintaining software?

A low quality design will not only slow implementation, thus increasing expense and delay, but will also force design decisions to be revisited later. Inevitably, the design will have to change. Related implementation will have to be adapted or recoded. Testing will have to be redone. The more patchwork done, the more likely new errors are injected into the code. Not only are the maintenance programmers burdened with what is almost certainly a less coherent design (and documentation that tended to become outdated as more changes to the design were made), but they must also hunt for more bugs.