# Inheritance

## 1. `Employee` and `ProductionWorker` Classes

Design a class named `Employee`. The class should keep the following information in fields:

- Employee name
- Employee number in the format XXX–L, where each X is a digit within the range 0–9 and the L is a letter within the range A–M.
- Hire date

Write one or more constructors and the appropriate accessor and mutator methods for the class.

Next, write a class named `ProductionWorker` that extends the `Employee` class. The `ProductionWorker` class should have fields to hold the following information:

- Shift (an integer)
- Hourly pay rate (a double)

The workday is divided into two shifts: day and night. The shift field will be an integer value representing the shift that the employee works. The day shift is shift 1 and the night shift is shift 2. Write one or more constructors and the appropriate accessor and mutator methods for the class. Demonstrate the classes by writing a program that uses a `ProductionWorker` object.

* **Lời dịch**:

Thiết kế lớp `Employee` có các thuộc tính sau:

- Tên nhân viên
- Mã số nhân viên định dạng XXX-L, trong đó X là chữ số trong phạm vi [0-9], M là chữ cái nằm trong phạm vi [A-M]
- Ngày vào làm

Viết một hoặc nhiều hàm khởi tạo cho lớp `Employee` với các phương thức kiểm tra các thuộc tính trong lớp hợp lệ.

Viết tiếp lớp `ProductionWorker` kế thừa lớp Employee. Lớp `ProductionWorker` có các thuộc tính:

- Ca làm việc (kiểu integer)
- Tiền công phải trả trên 1 giờ ( kiểu double)

Ngày làm việc được chia thành hai ca: ngày và đêm. Thuộc tính ca làm việc (shift) sẽ là một giá trị nguyên đại diện cho ca (shift) mà nhân viên đó làm việc. Ca ngày là ca 1 và ca đêm là ca 2. Viết

một hoặc nhiều hàm khởi tạo và các phương thức truy cập các thuộc tính của lớp .Viết một chương trình có hàm `main()` để kiểm thử đối tượng `ProductionWorker`

## 2. `ShiftSupervisor` Class

In a particular factory, a shift supervisor is a salaried employee who supervises a shift. In addi-tion to a salary, the shift supervisor earns a yearly bonus when his or her shift meets produc-tion goals. Design a ShiftSupervisor class that extends the Employee class you created in Programming Challenge 1. The ShiftSupervisor class should have a field that holds the annual salary and a field that holds the annual production bonus that a shift supervisor has earned. Write one or more constructors and the appropriate accessor and mutator methods for the class. Demonstrate the class by writing a program that uses a ShiftSupervisor object.

* **Lời dịch**:

Lớp `ShiftSupervisor`

Trong một phân xưởng, trưởng nhóm quản lý một ca làm việc là một nhân viên làm công ăn lương. Ngoài tiền lương, trưởng nhóm sẽ có một khoản tiền thưởng hàng nếu phân xưởng đạt được mục tiêu về sản lương.

Thiết kế một lớp `ShiftSupervisor` kế thừa lớp Employes (lớp này được tạo ra ở câu 1). Lớp `ShiftSupervisor` có thuộc tính lương (salary) và tiền thưởng (bonus) để lưu thông tin trưởng nhóm làm việc kiếm được tiền thưởng. Viết một hoặc nhiều khởi hàm tạo và các phương thức truy cập thuộc tính. Viết một chương trình có hàm `main()` để kiểm thử đối tượng `ShiftSupervisor`.

## 3. `TeamLeader` Class

In a particular factory, a team leader is an hourly paid production worker that leads a small team. In addition to hourly pay, team leaders earn a fixed monthly bonus. Team leaders are required to attend a minimum number of hours of training per year. Design a TeamLeader class that extends the ProductionWorker class you designed in Programming Challenge 1. The TeamLeader class should have fields for the monthly bonus amount, the required num-ber of training hours, and the number of training hours that the team leader has attended. Write one or more constructors and the appropriate accessor and mutator methods for the class. Demonstrate the class by writing a program that uses a TeamLeader object.

* **Lời dịch**:

Trong một phân xưởng, trưởng nhóm là một công nhân sản xuất được trả lương theo giờ, lãnh đạo một nhóm nhỏ. Ngoài tiền lương theo giờ, các trưởng nhóm còn kiếm được một khoản tiền thưởng cố định hàng tháng. Các trưởng nhóm được yêu cầu tham gia số giờ đào tạo tối thiểu mỗi năm. Thiết kế lớp `TeamLeader` kế thừa lớp `ProductionWorker` mà bạn đã thiết kế ở câu 1. Lớp `TeamLeader` phải có các trường về số tiền thưởng hàng tháng (nonthlyBonus), số giờ đào tạo bắt buộc (requiredTrainingHours) và số giờ đào tạo mà trưởng nhóm đã tham gia (trainingHoursAttended) . Viết một hoặc nhiều hàm khởi tạo và các phương thức truy cập và các

phương thức thích hợp cho lớp ( nếu có). Viết một chương trình có hàm `main()` để kiểm thử đối tượng `TeamLeader`.

### 4. `Essay` Class

Design an Essay class that extends the GradedActivity class presented in this chapter. The Essay class should determine the grade a student receives for an essay. The student's essay score can be up to 100 and is determined in the following manner:

Grammar: 30 points

Spelling: 20 points

Correct length: 20 points

Content: 30 points

Demonstrate the class in a simple program.

* **Lời dịch**:

Thiết kế một lớp `Essay` kế thừa lớp `GradedActivity` được giới thiệu trong chương 10. Lớp `Essay` phải xác định điểm mà học sinh nhận được cho một essay. Điểm bài essay của sinh viên có thể lên đến 100 và được xác định theo cách sau:

Ngữ pháp: 30 điểm

Chính tả: 20 điểm

Độ dài bài luận: 20 điểm

Nội dung: 30 điểm

Tạo lớp có hàm `main()` để kiểm thử chương trình

### 5. Course Grades

In a course, a teacher gives the following tests and assignments:

- A lab activity that is observed by the teacher and assigned a numeric score.
- A pass/fail exam that has 10 questions. The minimum passing score is 70.
- An essay that is assigned a numeric score.
- A final exam that has 50 questions.

Write a class named CourseGrades. The class should have a GradedActivity array named grades as a field. The array should have four elements, one for each of the assignments previously described. The class should have the following methods:

| | |
|---|---|
| setLab: | This method should accept a GradedActivity object as its argument. This object should already hold the student's score for the lab activity. Element 0 of the grades field should reference this object. |
| setPassFailExam: | This method should accept a PassFailExam object as its argument. This object should already hold the student's score for the pass/fail exam. Element 1 of the grades field should reference this object. |
| setEssay: | This method should accept an Essay object as its argument. (See Programming Challenge 4 for the Essay class. If you have not completed Programming Challenge 4, use a GradedActivity object instead.) This object should already hold the student's score for the essay. Element 2 of the grades field should reference this object. |
| setFinalExam: | This method should accept a FinalExam object as its argument. This object should already hold the student's score for the final exam. Element 3 of the grades field should reference this object. |
| toString: | This method should return a string that contains the numeric scores and grades for each element in the grades array. |

Demonstrate the class in a program.

* **Lời dịch**:

Trong một lớp học, giáo viên cho các bài kiểm tra và bài tập sau:

- Một bài thực thành trong phòng lab được giáo viên chấm điểm sau khi kết thúc bài thực hành.
- Một bài kiểm tra đạt/không đạt (pass/fail) có 10 câu hỏi. Điểm đạt tối thiểu là 70 điểm.
- Một bài luận (essay) thang điểm số
- Một bài thi cuối kỳ có 50 câu hỏi.

Viết một lớp có tên `CourseGrades`. Trong lớp này thuộc tính grades có kiểu dữ liệu là một mảng đối tượng GradedActivity. Mảng grades có 4 phần tử, mỗi phần tử đại diện cho một nhiệm vụ tương ứng được mô tả trước đó. Lớp có các phương thức sau:

| | |
|---|---|
| `setLab(GradedActivity aLab)` | Tham số `aLab` đại diện cho bài thực hành, được gán cho phần tử **0** trong mảng `grades[]` |
| `setPassFailExam(PassFailExam aPassFailExam)` | Tham số `aPassFailExam` đại diện cho bài kiểm tra pass/fail, được gán cho phần tử **1** trong mảng `grades[]` |
| `setEssay(Essay anEssay)` | Tham số anEssay đại diện cho điểm bài luận, được gán cho phần tử **2** trong mảng `grades[]` |
| `setFinalExam(FinalExam aFinalExam)` | Tham số aFinalExam đại diện cho điểm bài thi cuối kyỳ, được gán cho phần tử **3** trong mảng `grades[]` |
| `toString:` | Trả về một chuỗi chứa điểm số và điểm số cho mỗi phần tử trong mảng điểm (grade[]). |

Tạo lớp có hàm `main()` để kiểm thử chương trình

## 6. `Analyzable` Interface

Modify the `CourseGrades` class you created in Programming Challenge 5 so it implements the following interface:

```
public interface Analyzable
{
   double getAverage();
   GradedActivity getHighest();
   GradedActivity getLowest();
}
```

The `getAverage` method should return the average of the numeric scores stored in the `grades` array. The getHighest method should return a reference to the element of the grades array that has the highest numeric score. The `getLowest` method should return a reference to the element of the grades array that has the lowest numeric score. Demonstrate the new methods in a complete program.

* **Lời dịch**:

Cập nhật lớp `CourseGrades` trong bài 5 để nó implement giao diện (interface) sau:

```
public interface Analyzable
{
   double getAverage();
   GradedActivity getHighest();
   GradedActivity getLowest();
}
```

Phương thức `getAverage` sẽ trả về giá trị đểm trung bình được lưu trữ trong mảng `grades[]`. Phương thức `getHighest` sẽ trả về một tham chiếu đến phần tử của mảng `grades[]` có điểm số cao nhất. Phương thức `getLowest` sẽ trả về một tham chiếu đến phần tử của mảng `grades[]` có điểm số thấp nhất. Tạo lớp có hàm `main()` để kiểm thử chương trình

## 7. `Person` and `Customer` Classes

Design a class named `Person` with fields for holding a person's name, address, and telephone number. Write one or more constructors and the appropriate mutator and accessor methods for the class's fields.

Next, design a class named `Customer`, which extends the `Person` class. The `Customer` class should have a field for a customer number and a `boolean` field indicating whether the customer wishes to be on a mailing list. Write one or more constructors and the appropriate mutator and accessor methods for the class's fields. Demonstrate an object of the `Customer` class in a simple program.

* **Lời dịch**:

Thiết kế class `Person` có các thuộc tính: name, address, và telephone number.

- Viết các hàm khởi tạo (Contructor)
- Viết getter và setter cho lớp này

Tiếp theo, thiết kế class `Customer` kế thừa class `Person`. Class `Customer` có các thuộc tính: customer number và mailing list kiểu `boolean`, mailing list là thuộc tính cho biết liệu khách hàng có muốn nằm trong danh sách gửi thư hay không.

## 8. `PreferredCustomer` Class

A retail store has a preferred customer plan where customers can earn discounts on all their purchases. The amount of a customer's discount is determined by the amount of the cus-tomer's cumulative purchases in the store as follows:

- When a preferred customer spends $500, he or she gets a 5 percent discount on all future purchases.
- When a preferred customer spends $1,000, he or she gets a 6 percent discount on all future purchases.
- When a preferred customer spends $1,500, he or she gets a 7 percent discount on all future purchases.
- When a preferred customer spends $2,000 or more, he or she gets a 10 percent dis-count on all future purchases.

Design a class named `PreferredCustomer`, which extends the `Customer` class you created in Programming Challenge 7. The `PreferredCustomer` class should have fields for the amount of the customer's purchases and the customer's discount level. Write one or more construc-tors and the appropriate mutator and accessor methods for the class's fields. Demonstrate the class in a simple program.

* **Lời dịch**:

Một cửa hàng bán lẻ có kế hoạch khách giảm giá cho khách mua hàng. Số tiền giảm giá của khách hàng được xác định bằng số lượng mua sắm tích lũy của khách hàng trong cửa hàng như sau:

- Khi số tiền mua hàng là 500 đô la, được giảm 5% cho các lần mua hàng tiếp theo.
- Khi số tiền mua hàng là 1000 đô la, được giảm 6% cho các lần mua hàng tiếp theo.
- Khi số tiền mua hàng là 1500 đô la, được giảm 7% cho các lần mua hàng tiếp theo.
- Khi số tiền mua hàng từ 2000 đô la, được giảm 10% cho các lần mua hàng tiếp theo.

Thiết kế một lớp có tên `PreferredCustomer` thừa kế từ class `Customer` trong câu 7. Lớp `PreferredCustomer` có các thuộc tính: amount (số tiền mua hàng) và discount level (mức giảm giá).

- Viết các Constructor
- Viết Getter và Setter cho lớp này
- Viết phương thức tính tiền giảm giá dựa vào số tiền
- Gọi và sử dụng các class vừa viết

## 9. `BankAccount` and `SavingsAccount` Classes

Design an abstract class named `BankAccount` to hold the following data for a bank account:

- Balance
- Number of deposits this month
- Number of withdrawals
- Annual interest rate
- Monthly service charges

The class should have the following methods:

| | |
|---|---|
| Constructor: | The constructor should accept arguments for the balance and annual interest rate. |
| `deposit`: | A method that accepts an argument for the amount of the deposit. The method should add the argument to the account balance. It should also increment the variable holding the number of deposits. |
| `withdraw`: | A method that accepts an argument for the amount of the withdrawal. The method should subtract the argument from the balance. It should also increment the variable holding the number of withdrawals. |
| `calcInterest`: | A method that updates the balance by calculating the monthly interest earned by the account, and adding this interest to the balance. This is performed by the following formulas:<br>*Monthly Interest Rate* = (*Annual Interest Rate* / 12)<br>*Monthly Interest* = *Balance* * *Monthly Interest Rate*<br>*Balance* = *Balance* + *Monthly Interest* |
| `monthlyProcess`: | A method that subtracts the monthly service charges from the balance, calls the `calcInterest` method, and then sets the variables that hold the number of withdrawals, number of deposits, and monthly service charges to zero. |

Next, design a `SavingsAccount` class that extends the `BankAccount` class. The `SavingsAccount` class should have a status field to represent an active or inactive account. If the balance of a savings account falls below $25, it becomes inactive. (The status field could be a boolean variable.) No more withdrawals may be made until the balance is raised above $25, at which time the account becomes active again. The savings account class should have the following methods:

| | |
|---|---|
| `withdraw:` | A method that determines whether the account is inactive before a withdrawal is made. (No withdrawal will be allowed if the account is not active.) A withdrawal is then made by calling the superclass version of the method. |
| `deposit:` | A method that determines whether the account is inactive before a deposit is made. If the account is inactive and the deposit brings the balance above $25, the account becomes active again. A deposit is then made by calling the superclass version of the method. |
| `monthlyProcess:` | Before the superclass method is called, this method checks the number of withdrawals. If the number of withdrawals for the month is more than 4, a service charge of $1 for each withdrawal above 4 is added to the superclass field that holds the monthly service charges. (Don't forget to check the account balance after the service charge is taken. If the balance falls below $25, the account becomes inactive.) |

* **Lời dịch**:

Thiết kế lớp trừu tượng `BankAccount` để lưu trữ thông tin về tài khoản ngân hàng như sau:

- Balance (Số tiền)
- Number of deposits this month (số tiền nạp vô trong tháng)
- Number of withdrawals (số tiền rút ra)
- Annual interest rate (lãi suất hàng năm)
- Monthly service charges (phí dịch vụ hàng tháng)

Các phương thức:

- Constructor
- Getter và Setter

| | |
|---|---|
| Constructor: | The constructor should accept arguments for the balance and annual interest rate. |
| `deposit:` | A method that accepts an argument for the amount of the deposit. The method should add the argument to the account balance. It should also increment the variable holding the number of deposits. |
| `withdraw:` | A method that accepts an argument for the amount of the withdrawal. The method should subtract the argument from the balance. It should also increment the variable holding the number of withdrawals. |
| `calcInterest:` | A method that updates the balance by calculating the monthly interest earned by the account, and adding this interest to the balance. This is performed by the following formulas:<br>*Monthly Interest Rate* = (*Annual Interest Rate* / 12)<br>*Monthly Interest* = *Balance* * *Monthly Interest Rate*<br>*Balance* = *Balance* + *Monthly Interest* |
| `monthlyProcess:` | A method that subtracts the monthly service charges from the balance, calls the `calcInterest` method, and then sets the variables that hold the number of withdrawals, number of deposits, and monthly service charges to zero. |

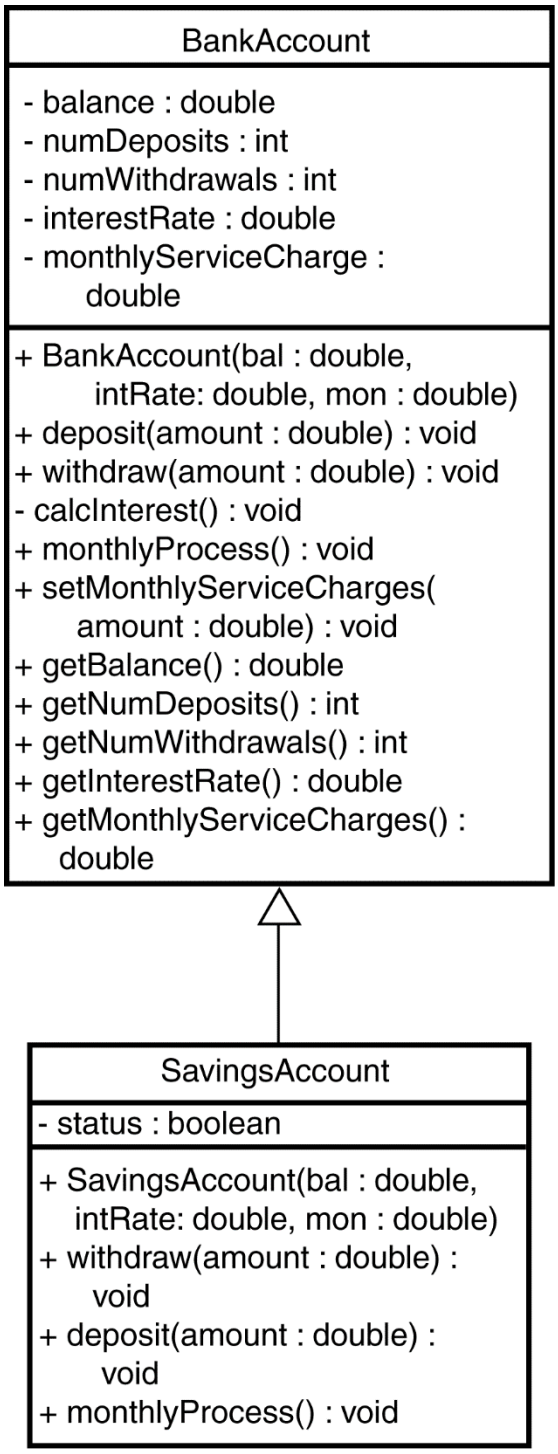
Tiếp theo, thiết kế lớp `SavingsAccount` thừa kế lớp `BankAccount`. Lớp `SavingsAccount` có trường status để biểu diễn tài khoản này còn hoạt động hay không. Nếu số tiền (balance) nhỏ hơn 25 thì coi như tài khoản này không còn hoạt động. (Trường status nên

là kiểu boolean). Tiền chỉ được rút khi số tiền lớn hơn 25, có nghĩa là tài khoản đang hoạt động. Các phương thức như sau:

- Các phương thức khởi tạo (Contructor)
- Các phương thức getter và getter

| | |
|---|---|
| `withdraw:` | A method that determines whether the account is inactive before a withdrawal is made. (No withdrawal will be allowed if the account is not active.) A withdrawal is then made by calling the superclass version of the method. |
| `deposit:` | A method that determines whether the account is inactive before a deposit is made. If the account is inactive and the deposit brings the balance above $25, the account becomes active again. A deposit is then made by calling the superclass version of the method. |
| `monthlyProcess:` | Before the superclass method is called, this method checks the number of withdrawals. If the number of withdrawals for the month is more than 4, a service charge of $1 for each withdrawal above 4 is added to the superclass field that holds the monthly service charges. (Don't forget to check the account balance after the service charge is taken. If the balance falls below $25, the account becomes inactive.) |

Sơ đồ UML: `BankAccount` và `SavingsAccount`




### 10. `Ship`, `CruiseShip`, and `CargoShip` Classes

Design a `Ship` class that the following members:

- A field for the name of the ship (a `string`).
- A field for the year that the ship was built (a `string`).

- A constructor and appropriate accessors and mutators.
- A `toString` method that displays the ship's name and the year it was built.

Design a `CruiseShip` class that extends the `Ship` class. The `CruiseShip` class should have the following members:

- A field for the maximum number of passengers (an `int`).
- A constructor and appropriate accessors and mutators.
- A `toString` method that overrides the `toString` method in the base class. The `CruiseShip` class's `toString` method should display only the ship's name and the maximum number of passengers.

Design a `CargoShip` class that extends the `Ship` class. The `CargoShip` class should have the following members:

- A field for the cargo capacity in tonnage (an `int`).
- A constructor and appropriate accessors and mutators.
- A `toString` method that overrides the `toString` method in the base class. The `CargoShip` class's `toString` method should display only the ship's name and the ship's cargo capacity.

Demonstrate the classes in a program that has a `Ship` array. Assign various `Ship`, `CruiseShip`, and `CargoShip` objects to the array elements. The program should then step through the array, calling each object's `toString` method. (See Code Listing 10-25 as an example.)