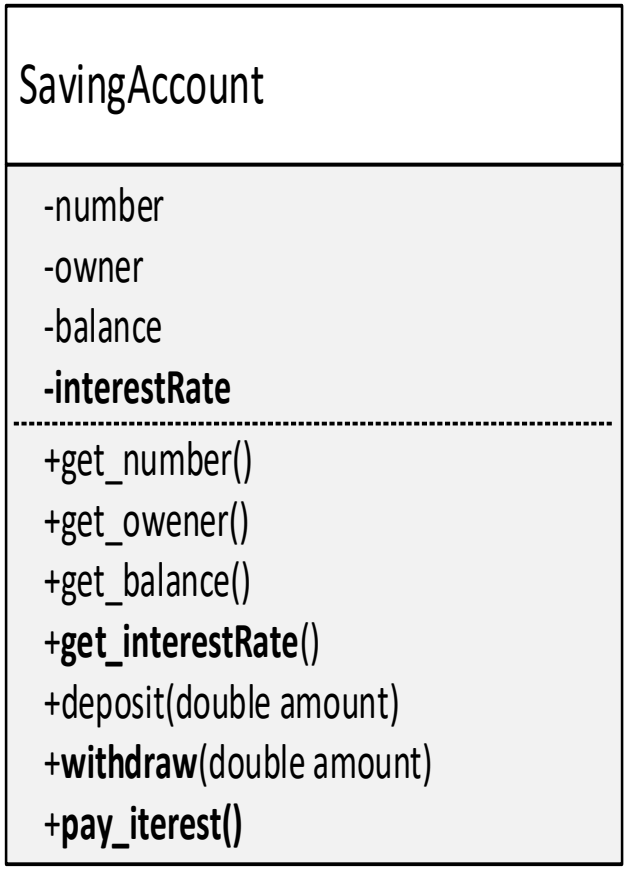
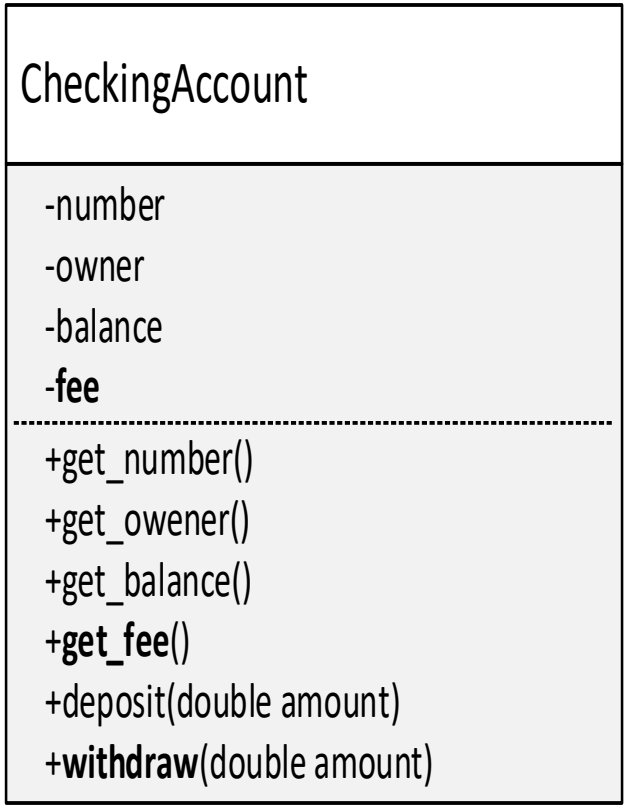
Object Inheritance

Humayun Kabir
Professor, CS, Vancouver Island University, BC, Canada

Classes with Common Properties

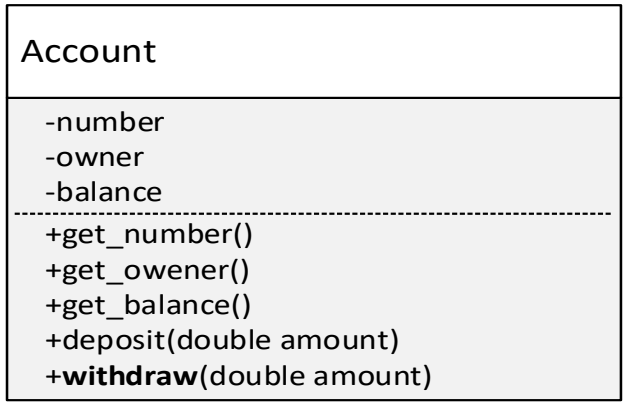


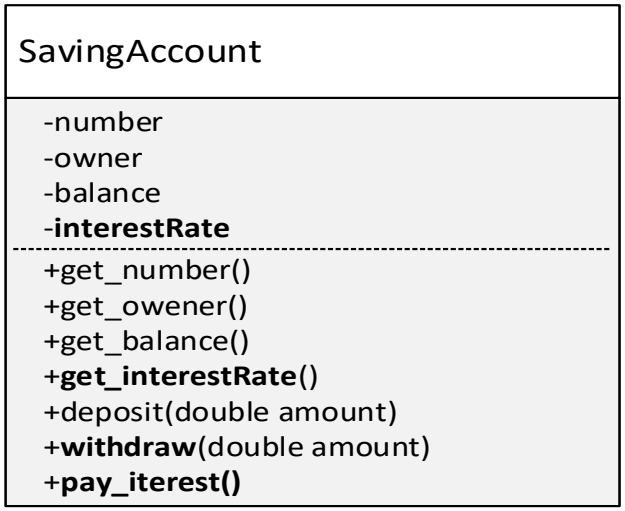


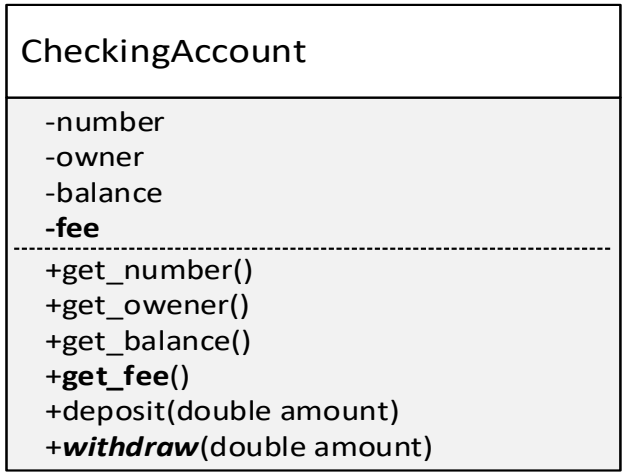
Inheritance

- Classes may be arranged in a class hierarchy where one class (**base class**) is a generalisation of one or more other classes (**derived classes**).
- A derived class **inherits** the **attributes** and **behaviours** from its base class and **may add new operations** or **attributes** of its own.
- Generalisation is implemented as inheritance in OOP languages.

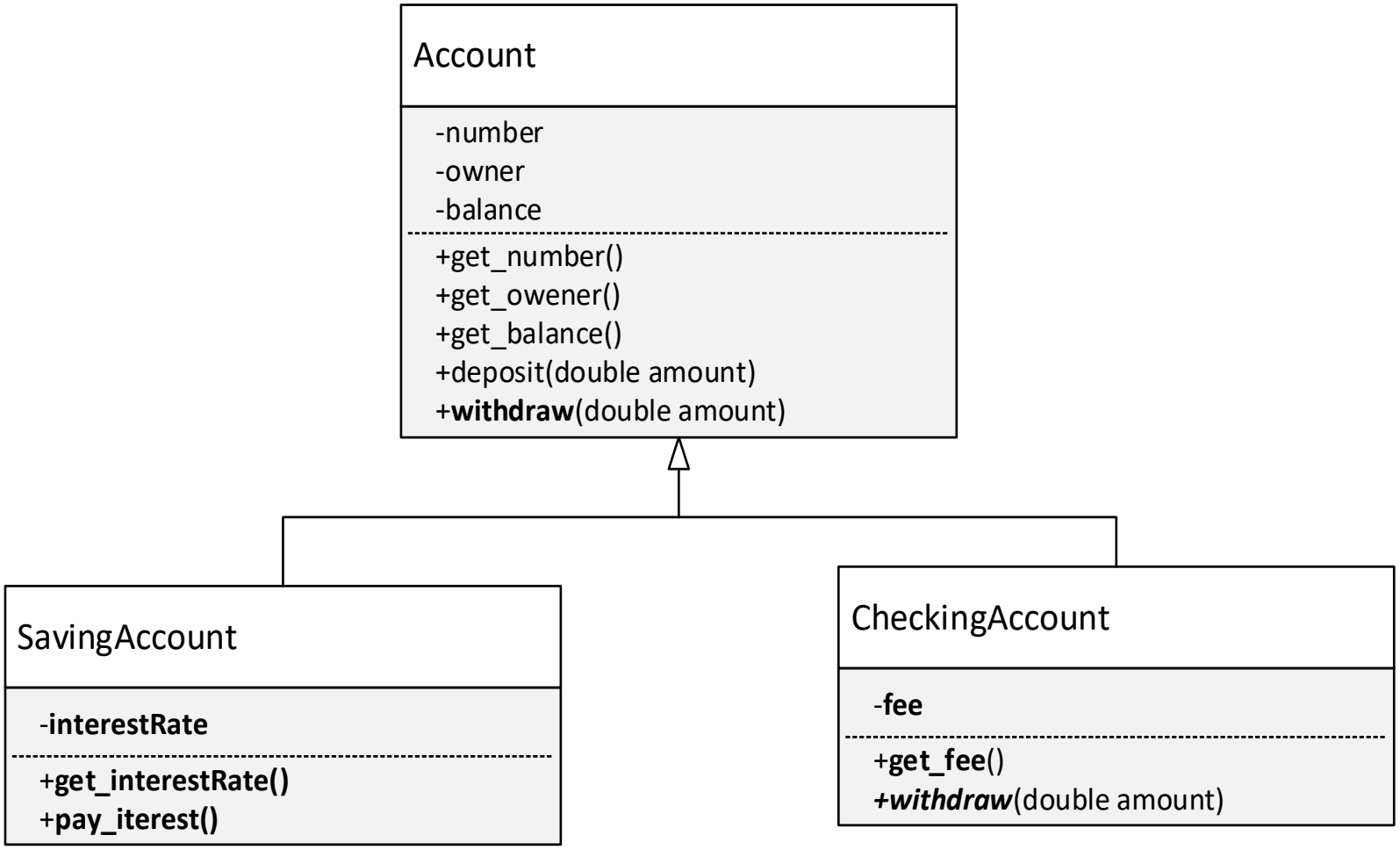
Class Hierarchy







Class Inheritance Hierarchy



Base class and Derived class

- The advantage of making a new class a derived class is that it will *inherit* attributes and operations of its base class.
- Derived classes extend existing classes in three ways:
  - By defining new (additional) attributes and operations.
  - By overriding (changing the behavior) existing operations.
  - By hiding existing attributes and operations.

Derived classes

- Derived classes are used to define special cases, extensions, or other variations from the originally defined class.

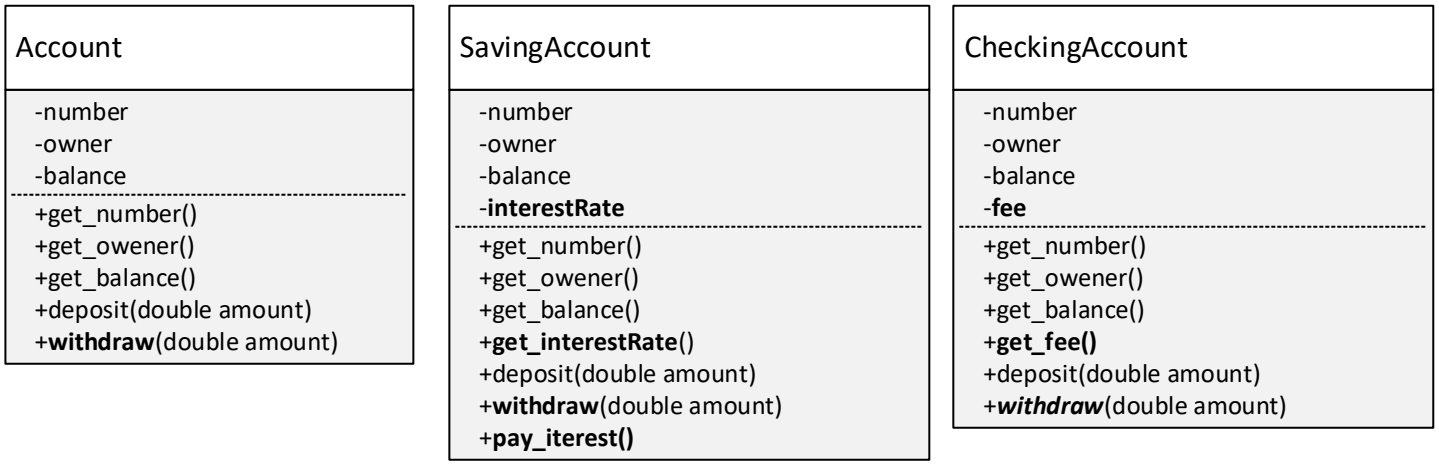
Example: ***SavingsAccount*** and ***CheckingAccount*** can be derived from the ***Account*** class

```
class Account {
  private:
            string number;
            string owner;
            double balance;
  public:
            string get_number() {…}
            string get_owner() {…}
            double get_balance() {…}
            void deposit(double) {…}
            void withdraw(double) {…}
};
```

```
class SavingsAccount : public Account {
   private:
            double interestRate;
   public:
            double get_interestRate() {...}
            void pay_interest(double) {…}
};
```

```
class CheckingAccount: public Account {
   private:
            double fee;
   public:
            double get_fee() {…}
            void withdraw(double) {…}
};
```

Derived classes



- No new code written for **get_number()**, **get_owner()**, **get_balance()**, and **deposit()** operations, they are **inherited** from the **base class** into the **derived classes**.
- *SavingsAccount* inherited **withdraw*()*** operation and added new **get_interestR*ate()*** and **pay_interest()** operations.
- ***CheckingAccount*** has different rules to follow for withdraw, i.e, overrode ***withdraw()*** operation and added new **get_fee**() operation.

Inheritance: Rules

- Derived class **inherits** but **cannot access private member variables**
- Derived class does not inherit **private member functions**.
- Derived class **inherits** every **non-private member variables** and **non-private member functions** except:
  - **Constructors** (default, regular, copy, and move)
  - **Destructor**
  - **Assignment operators** (copy assignment and move assignment)
- Derived class does not inherit **friend functions**

Advantages of Inheritance

- It is a **reuse mechanism** at both the design and the programming level
- It is an **abstraction mechanism** which may be used to classify entities