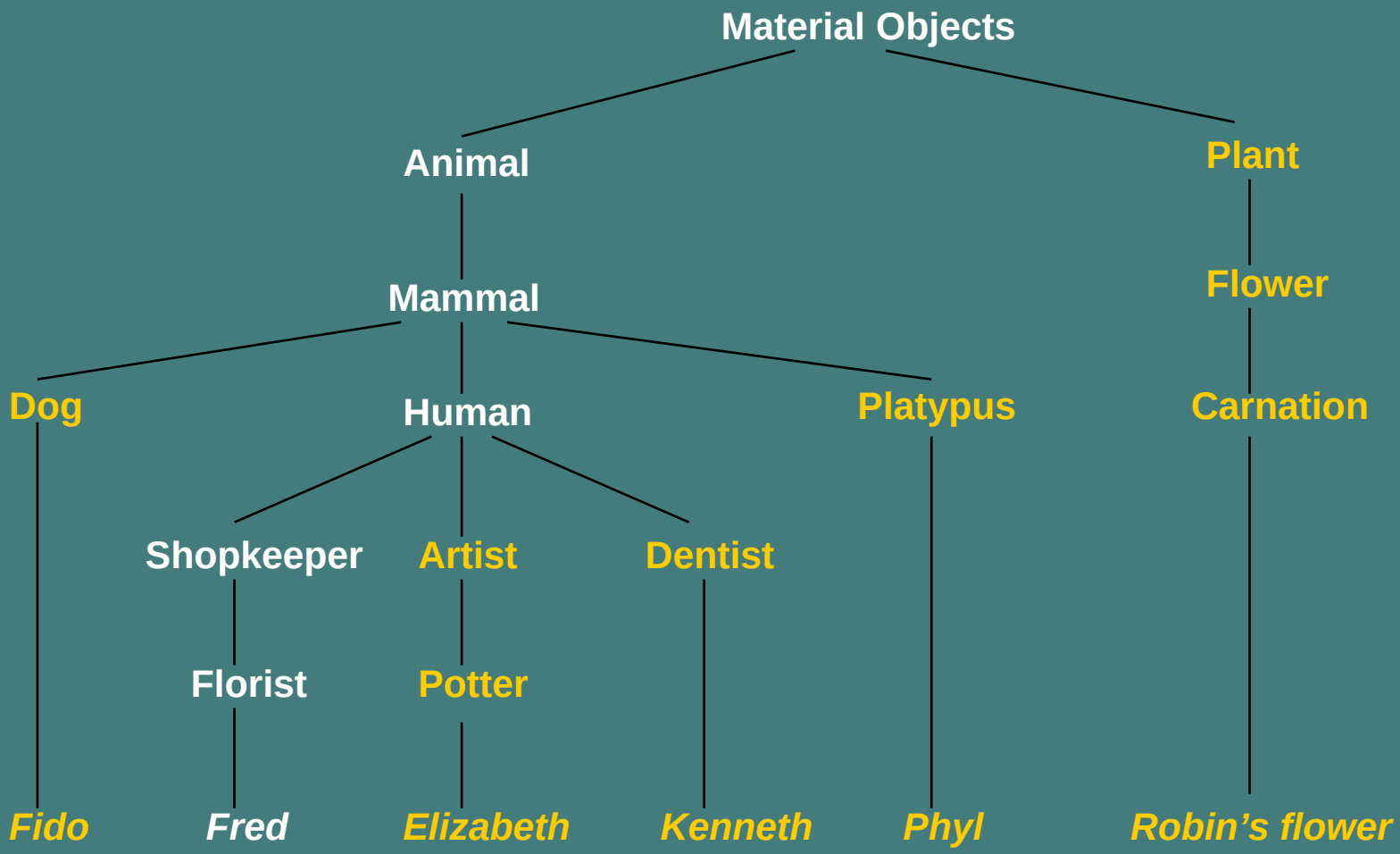
# Chapter 11

# Inheritance

# Chapter 11 Topics(part 1)

- Why use inheritance?
- Derivation（派生）
  - Private Inheritance
  - Public Inheritance
- Protected Member
  - Visibility Modifiers （可见性修饰符）
  - Visibility in Public Inheritance
  - Visibility in Private Inheritance

# Abstract idea of Inheritance



# The *is-a* Test

Try forming the English sentences "An A *is-a* B". If it "sounds right" to your ear, then A can be made a **subclass(子类)** of B.

## EXAMPLES

- A cat *is-a* mammal, and therefore a cat inherits from mammal
- A bird *is-a* mammal sounds wrong, and therefore inheritance is not natural.

# Practical Meaning of Inheritance

- Data members in the parent are part of the child
- Behavior defined in the parent are part of the child

# Extension(扩展) and Contraction(收缩)

- Because the behavior of a child class is strictly larger than the behavior of the parent, the child is an extension of the parent. (larger)
- Because the child can override（改写） behavior to make it fit a specialized situation, the child is a contraction of the parent. (smaller)

# Reasons to Use Inheritance

- **Reuse of code. Methods defined in the parent can be made available to the child without rewriting. Makes it easy to create new abstractions.**
- **Reuse of concept. Methods described in the parent can be redefined and overridden in the child. Although no code is shared between parent and child, the concept embodied(包含) in the definition is shared.**

# Chapter 11 Topics(part 1)

- Why use inheritance?
- Derivation
  - Private Inheritance
  - Public Inheritance
- Protected Member
  - Visibility Modifiers
  - Visibility in Public Inheritance
  - Visibility in Private Inheritance

# Derived Class Declaration

```
class ChildClassName:public ParentClassName{
...
};
```

OR

```
class ChildClassName:[private] ParentClassName{
...
};
```

# Private Inheritance

- All the members in the parent class become private members in the child class.
- The inherited public members in the parent class can be accessed within the child class.
- The inherited private members in the parent class can not be accessed directly within the child class.
- The external function can not access the inherited members through child class.

# Example

```
class Parent{
public:
    int one;
    Parent( ){one=10; two=10;}
    void inParent( ){cout<<one<<two;}
private:
    int two;
};
```

```
class Child:private Parent{
public:
    void inChild( ){
        cout<<one;  // legal
        cout<<two;   // illegal
    }
};
```

```
int main( ){
Child c;
cout<<c.one; // illegal
cout<<c.two; // illegal
return 0;
}
```

# Public Inheritance

- All the members in the parent class remain original states in the child class.
- The inherited public members in the parent class can be accessed within the child class.
- The inherited private members in the parent class can not be accessed directly within the child class.
- The external function can only access the inherited public members through child class.

# Example

```
class Parent{
public:
    int one;
    Parent( ){one=10; two=10;}
    void inParent( ){cout<<one<<two;}
private:
    int two;
};
```

```
class Child:public Parent{
public:
    void inChild( ){
        cout<<one;  // legal
        cout<<two;   // illegal
    }
};
```

```
int main( ){
Child c;
cout<<c.one; // legal
cout<<c.two; // illegal
return 0;
}
```

# Chapter 11 Topics(part 1)

- Why use inheritance?
- Derivation
  - Private Inheritance
  - Public Inheritance
- Protected Member
  - Visibility Modifiers
  - Visibility in Public Inheritance
  - Visibility in Private Inheritance

# Class Declaration

## SYNTAX

```
class ClassName {
public:
        //public data members and member functions
       DataType   MemberName ;
            .
            .
            .
protected:
        //protected data members and member functions
       DataType   MemberName ;
            .
            .
            .
[private:]
        //private data members and member functions
       DataType   MemberName ;
            .
            .
            .
};
```

# Visibility Modifiers

- **private**: accessible only within the class definition (but memory is still found in the child class, just not accessible).
- **public**: accessible anywhere
- **protected**: accessible within the class definition and within the definition of child classes.

# Example

```
class Parent{
public:
    int one;
    Parent( ){one=10;two=10;three=10;}
    void inParent( ){cout<<one<<two<<three;}
protected:
    int three;
private:
    int two;
};
```

```
class Child:public Parent{
public:
    void inChild( ){
        cout<<one;  // legal
        cout<<two;  // illegal
        cout<<three;  // legal
    }
};
```

```
int main( ){
Child c;
cout<<c.one; // legal
cout<<c.two;  // illegal
cout<<c.three;  // illegal
return 0;
}
```

# Visibility in Public Inheritance

| members in parent class | private | protected | public |
|---|---|---|---|
| child class | not accessible | accessible | accessible |
| external function | not accessible | not accessible | accessible |

# Visibility in Private Inheritance

| members in parent class | private | protected | public |
|---|---|---|---|
| child class | not accessible | accessible | accessible |
| external function | not accessible | not accessible | not accessible |