Lecture 3

Inheritance -Part 1

University of Anbar
College of Computer Science and Information Technology
Department of Computer Science
Object Oriented Programming
Second Class
Dr. Ruqayah R. Al-Dahhan

Inheritance

- One of the most **important** concepts in **object-oriented programming** is that of **inheritance**. Inheritance allows us to **define a class in terms of another class**, which makes it easier to create and maintain an application. This also provides an opportunity to **reuse the code** functionality and **fast implementation** time
- We frequently **classify** objects.

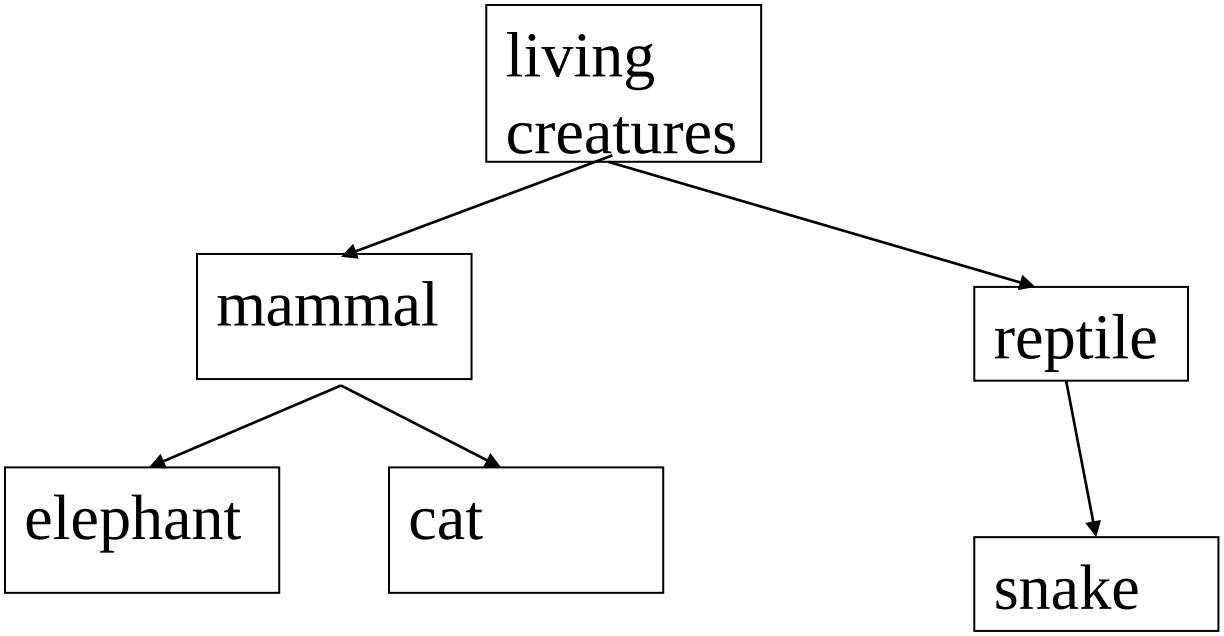
Mammals are

- **warm-blooded**
- **higher vertebrates.**

This information is valid for elephant and mouse but is best if we only have to express it only once for all mammals and not have to duplicate for every mammal.

Large body of knowledge can be presented in a compact way.

Inheritance



The class cat is **derived** from the class mammal.We say that a cat "**is a** mammal" and is also a "living creature". But a cat "is not a reptile". It is often useful to build our Object-Oriented programs this way.

Inheritance

- Inheritance is the capability of one class to acquire properties and characteristics from another class.
- The class whose properties are inherited by other class is called the **Parent** or **Base** or **Super** class.
- The class which inherits properties of other class is called **Child** or **Derived** or **Sub** class.

How to define a derived class?

- To define a derived class, we use a class derivation list to specify the base class(es). A class derivation list names one or more base classes and has the form:

***Class Derived**class: **Access-Specifier Base**class*

↑ ↑

Keyword ***(Public, Private, Protected)***

- If the access-specifier is not used, then it is private by default.

Example: Consider a base class **Shape** and its derived class **Rectangle** as follows:

```
#include <iostream>
 using namespace std;
// Base class
class Shape {
   public:
      void setWidth(int w) {
         width = w;
      }
      void setHeight(int h) {
         height = h;
      }
    protected:
      int width;
      int height;
};
// Derived class
class Rectangle: public Shape {
   public:
      int getArea() {
         return (width * height);
      }
};
main() {
   Rectangle Rect;
   Rect.setWidth(5);
   Rect.setHeight(7);

   // Print the area of the object.
   cout << "Total area: " << Rect.getArea() <<
endl;

   return 0;
}
```

Notes

- The **Output:** When the previous code is compiled and executed, it produces the following result:

  Total area: 35

- The public inheritance relations between the classes:
  - Shape is the base class, and Rectangle is derived from it.

- We can group information (and avoid having to duplicate code) in a way that reflects our application and the real things or ideas that our code models.
- In main, a **Rect** object can call any functions in the Base class (**Shape**) that is derived from.