Programming ||

Spring, 2023

Musa Hodman

- Lecture 12
- Inheritance

Road Map

- C++ Inheritance
- Base & Derived Classes
- Access control and Inheritance
- Types of Inheritance

Inheritance

- Inheritance is one of the key features of Object-oriented programming in C++.
- It allows user to create a new class (derived class) from an existing class(base class).

- The derived class inherits all the features from the base class and can have additional features of its own.

- The idea of inheritance implements the **is a** relationship.
  - For example, mammal IS-A animal, dog IS-A mammal hence dog IS-A animal as well and so on.

Why inheritance should be used?

- Suppose, in your game, you want three characters - a **maths teacher**, a **footballer** and a **businessman**.

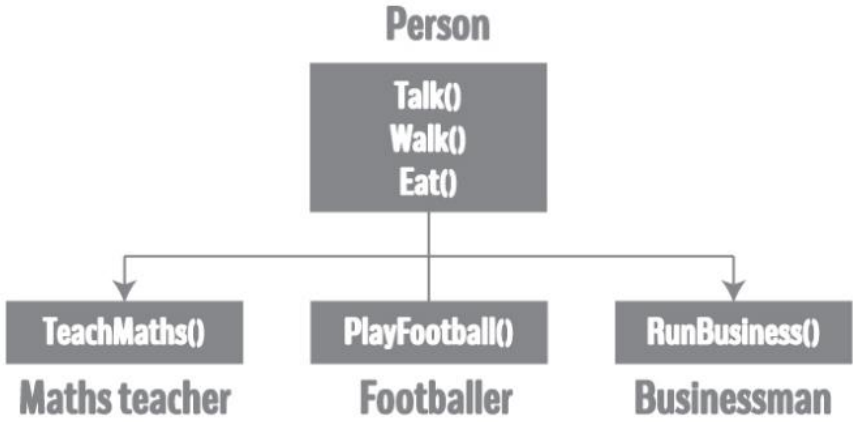
- Since, all of the characters are persons, they can walk and talk. However, they also have some special skills. A maths teacher can **teach maths**, a footballer can **play football** and a businessman can **run a business**.

- You can individually create three classes who can walk, talk and perform their special skill as shown in the figure below.

Cont…

- In each of the classes, you would be copying the same code for walk and talk for each character.
- If you want to add a new feature - eat, you need to implement the same code for each character. This can easily become error prone (when copying) and duplicate codes.

- **Solution->**



Access control and inhritance

- We can summarize the different access types according to who can access them in the following way:

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

- A derived class inherits all base class methods with the following exceptions:
  - Constructors, destructors and copy constructors of the base class.
  - Overloaded operators of the base class.
  - The friend functions of the base class.

Advantage of Inheritance

- Reusability of code
- Size of the code is reduced
- Transitivity
  - If B is derived from A and C is derived from B then C is also derived from AA.

Syntax

```
class Person
{
   ... .. ...
};

class MathsTeacher : public Person
{
   ... .. ...
};

class Footballer : public Person
{
   .... .. ...
};
```

```
class Person
{
      public:
         string profession;
         int age;

         Person(): profession("unemployed"), age(16) { }
         void display()
         {
              cout << "My profession is: " << profession << endl;
              cout << "My age is: " << age << endl;
              walk();
              talk();
         }
         void walk() { cout << "I can walk." << endl; }
         void talk() { cout << "I can talk." << endl; }
};

// MathsTeacher class is derived from base class Person.
class MathsTeacher : public Person
{
     public:
        void teachMaths() { cout << "I can teach Maths." << endl; }
};
```

```
// Footballer class is derived from base class Person.
class Footballer : public Person
{
    public:
       void playFootball() { cout << "I can play Football." << endl; }
};

int main()
{
     MathsTeacher teacher;
     teacher.profession = "Teacher";
     teacher.age = 23;
     teacher.display();
     teacher.teachMaths();

     Footballer footballer;
     footballer.profession = "Footballer";
     footballer.age = 19;
     footballer.display();
     footballer.playFootball();

     return 0;
}
```

Multiple Inheritances

- A C++ class can inherit members from more than one class and here is the extended syntax:

```
class derived-class: access baseA, access baseB....
```

Example

```
// Base class Shape
class Shape {
   public:
      void setWidth(int w) {
         width = w;
      }

      void setHeight(int h) {
         height = h;
      }

   protected:
      int width;
      int height;
};

// Base class PaintCost
class PaintCost  {
   public:
      int getCost(int area) {
         return area * 70;
      }
};
```

```
// Derived class
class Rectangle: public Shape, public PaintCost {
   public:
      int getArea() {
         return (width * height);
      }
};

int main(void) {
   Rectangle Rect;
   int area;

   Rect.setWidth(5);
   Rect.setHeight(7);

   area = Rect.getArea();

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

   // Print the total cost of painting
   cout << "Total paint cost: $"
   << Rect.getCost(area) << endl;

   return 0;
}
```

Any