Polymorphism

Contents

- Function overriding
- What is polymorphism
- Difference between compile time and run time binding.
- Need of virtual function.
- Types of classes

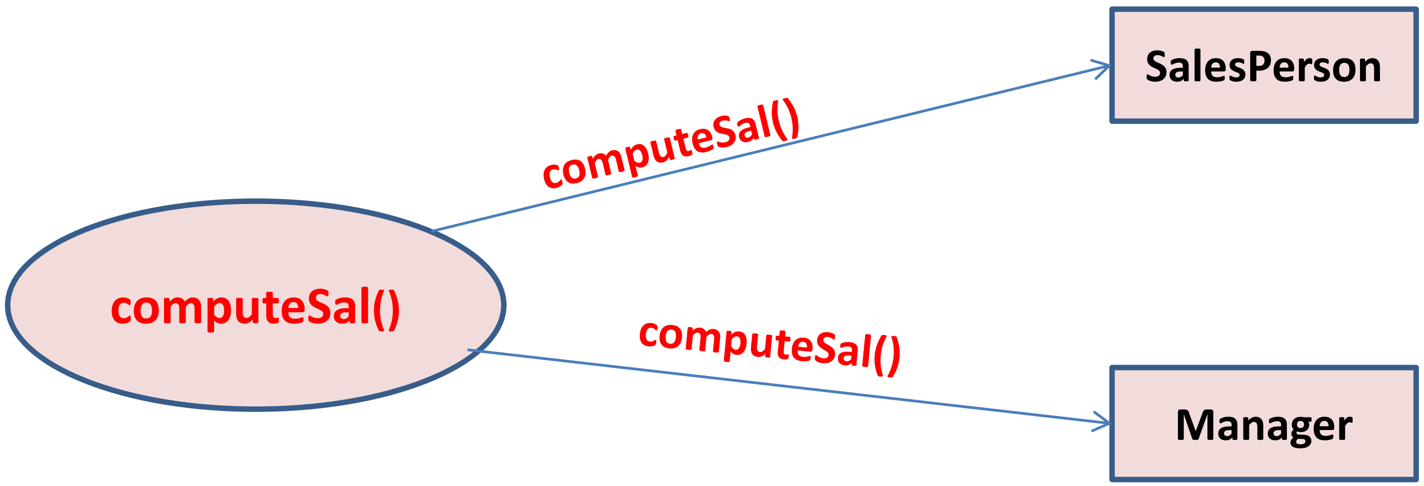
Function overriding

- All things are same like function name , function signature(function return type , function arguments).
- **For e.g.**

```
class BaseClass
{
		public:
				void show()
				{
						cout<<“Base class”;
				}
};
class DerivedClass : public BaseClass
{
		public:
		void show()
		{
				cout<<“Derived class”;
		}
};
```

Polymorphism

- The ability of different types of related objects to respond the same message in their own ways is called polymorphism.
- It helps to design extensible software.
- If new object are added then it will not affect your application.



What is compile and runtime binding

- Binding is an association of function call to an object.
- **Compile time binding**

  1. binding at compile time.

  2. Also called static binding or early binding.
- **Run time binding**

  1. binding at run time.

  2. Also called dynamic binding or late binding.

  3. Achieved by using virtual functions and inheritance.

Generic pointers

```
class BaseClass
{
        public:
        void show()
        {
                cout<<"Base class";
        }
};
class DerivedClass : public BaseClass
{
        public:
        void show()
        {
                cout<<"Derived class";
        }
};
```

```
int main( )
{
    BaseClass* ptr;
            DerivedClass d1;

    ptr=&d1;

    ptr -> show( );

}
```

Base class pointer

Here Compiler is unable to resolve function call. Compile time binding takes place.

How to resolve this problem???

- Need to make base class function as virtual.
- No need use virtual keyword for every function.
- when we make base class fi=unction as virtual the run time binding is applied.
- For e.g.

Used just virtual keyword

```
class BaseClass
{
            public:
            virtual void show()
            {
                        cout<<“Base class”;
            }
};
class DerivedClass : public BaseClass
{
            public:
            void show()
            {
                        cout<<“Derived class”;
            }
};
```

```
int main( )
{
    BaseClass* ptr;
    DerivedClass d1;

    ptr=&d1;

    ptr -> show( );

}
```

Here appropriate function is called.

Some points about Virtual Function

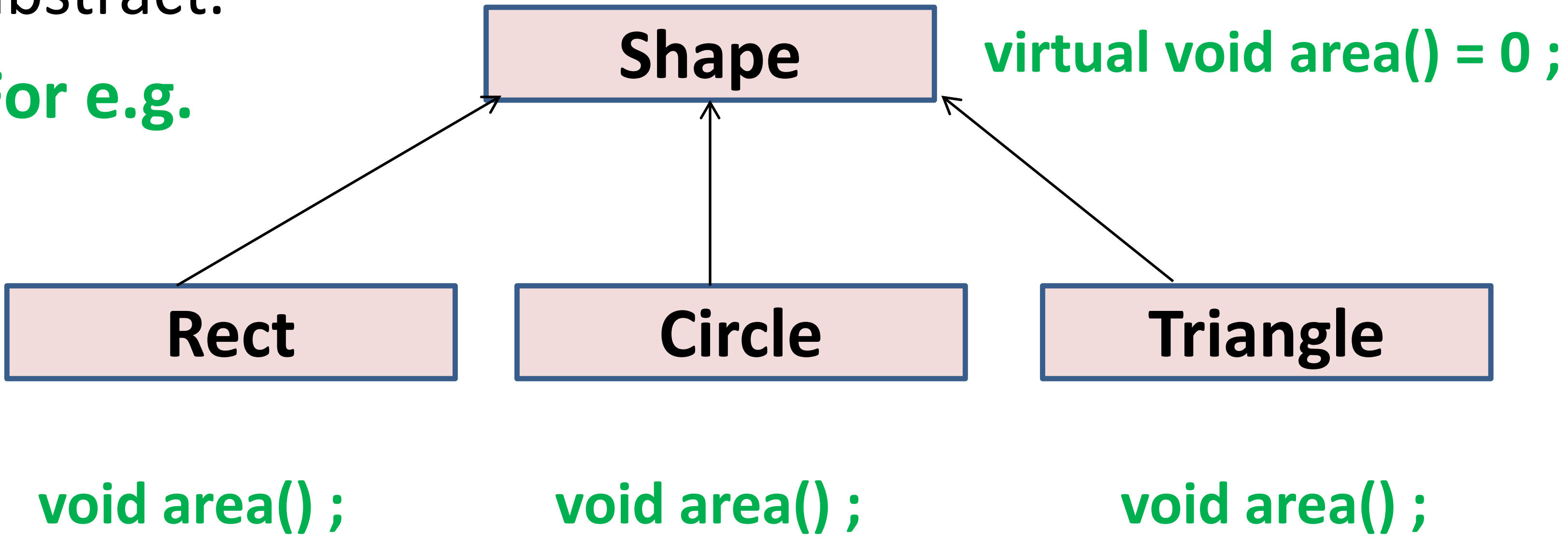
- Should be non-static member function of base class.
- Can not be used as friend function
- If function overridden then we can use virtual.
- Constructors can not be declared as virtual but destructors can be.
- If function is declared as virtual in the base class ,it will be treated as in derived class even if the virtual keyword is not used.

Pure virtual function

- A virtual function without any executable code.
- Declared by using a pure specifier (= 0) in the declaration of a virtual member function in the class declaration.
- Virtual float computeSal( ) = 0 ;
- A class at least one pure virtual function is termed as abstract class.

Abstract class

- An objet of abstract class can not be created.
- We can create pointer or reference.
- Pure virtual functions must be overridden in derived class otherwise derived classes are treated as also abstract.
- **For e.g.**



Types of classes

- **Concrete class**

  normal class.
- **Abstract class**

  Contains at least one pure virtual fun.
- **Pure abstract class**

  all functions are pure virtual function.
- **Polymorphic class**

  Contains at least one virtual function.

Lab assignments

- Create global function void show(cEmployee*)

And pass derived class object addresses to that function and by using common pointer name call to accept(),display() and computeSal().