Inheritance

A (base)
↑
B (derived)

```
class A
{
   // members
   // methods
};

class B : public A
{


};
```

Polymorphism

Pointers of the derived class are type-compatible with those of base class

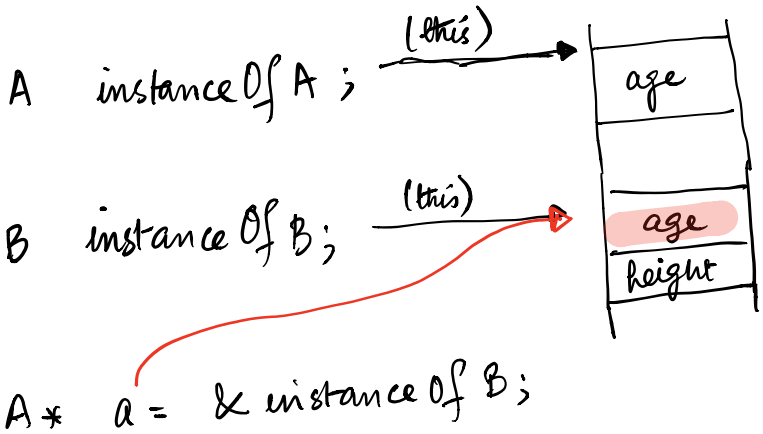
① `A *a = new B;`

②
```
B b;
A* a = &b;
```

③
```
B b
A& a = &b;
```

```
class A
{
   int age;
};
```

```
class B : class A
{
   int height;
};
```




```
class A {
void func1();
void func2();
};
```

```
class B : public A
void func1();
void func3();
};
```

over riding

```
A instance Of A;
instanceOf A. func1();
instance of A. func2();
instance of A. func3();
```

×

```
B instance Of B;
instance of B. func1();
instance of B. func2();
instance of B. func3();
```

A & a = instanceOf B;

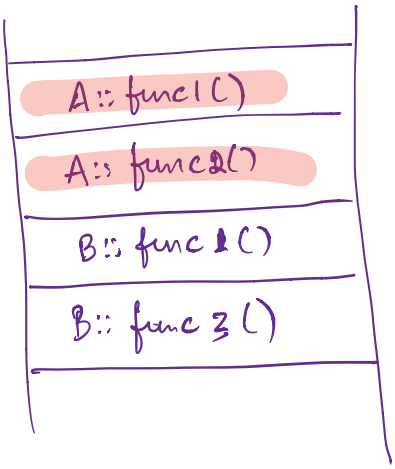
a.func1(); ⟶ func1 in A

a.func2(); ⟶ func2 in A

~~a.func3();~~ invalid

B instanceOf B;

A & a = instanceOf B;

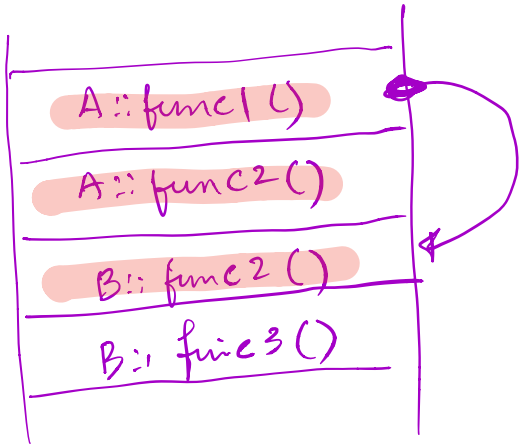



```
class A
{
  virtual void func1();
  void func2();
};
```

```
class B: public A
{
  void func1();
  void func3();
};
```

B instanceOf B;

A & a = instanceOf B;



Abstract Classes

1) A C++ class that has atleast 1 pure virtual function.
2) An instance to an Abstract class cannot be created.

```
class Shape
{
  public:
  virtual float area() = 0;   // a pure virtual function
};
```

① ~~Shape a;~~ not allowed

```
class Rectangle : public Shape
{
  protected:
  int height, width;
  public:
  Rectangle (int h, int w); height(h), width(w)
  { }

  float area() { return height * width; }

};
```

Now not an abstract class. ∵ It has implemented the missing bit.

Rectangle r;

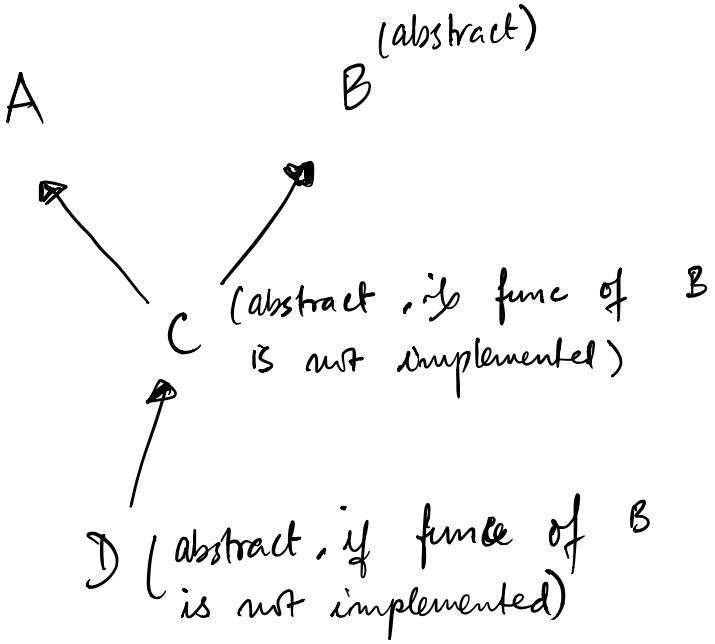
Inheritance

Polymorphism

Virtual functions → late binding
dynamic binding

Pure virtual functions

Abstract classes.



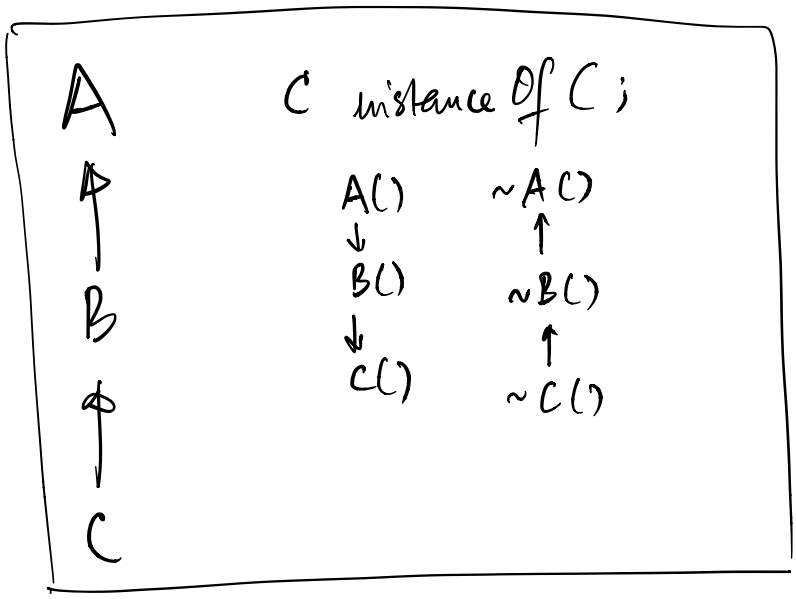
```
class A
{
    protected:
    int age;

    public:
    A(int a) : age(a). { }
```

```
Polygon
{
    ...

    Polygon (int a, int b);

};
Polygon :: Polygon (int a, int b)
{
    ...
}
```

```
class Rectangle :: public Polygon
{
    ...

};
```

~~`Rectangle r(10, 32);`~~ ✗

∵ Rectangle doesn't inherits the constructor from the parent class.



ASIDE

Rectangle r;

```
Rectangle :: Rectangle (int w, int h)
   : Polygon (w, h)
{

}
```

```
Rectangle :: Rectangle (int w, int h)
    : height(h), width(w) , Polygon()
{
  height = h;  width = w;
}
```