Class/Object Relationships

Downcasting

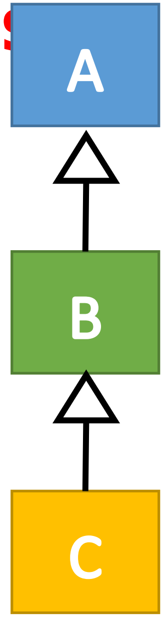
CS(217) Object Oriented Programming

Abeeda Akram

Inheritance (is-a) Down casting Pointers

- Down casting converts base class pointer to derived class pointer,
  - Only if base class pointer is pointing to derived class object.
  - `dynamic_cast` operator is used for down casting pointers
    - Determine object's type at runtime
    - Returns 0 or Null, if not of proper type (cannot be cast)
  - `dynamic_cast` will not work
    - With protected and private inheritance
    - With classes, which not have any virtual function.
- Down casting is helpful
  - For accessing explicitly derived class data and functions that does not exist in base class.

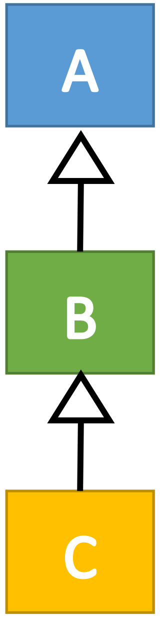
Inheritance (is-a) Downcasting Pointers

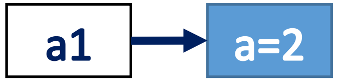


```
class A{
   int a;
public:
   A(int a=0){ this->a=a;}
   virtual void print(){ cout<<a;}
   virtual ~A(){}
};
class B: public A{
   int b;
public:
   B(int a=0, int b=0):A(a)
   { this->b = b;}
   void print() override{
   A::print();
   cout<<b;
   }
   virtual ~B(){}
};
```

```
class C: public B{
    int c;
public:
    C(int a=0, int b=0, int c=0) :B(a,b)
    { this->c = c;}
    void print() override{
        B::print();
        cout<<c;
    }
    virtual ~C(){}
};
```

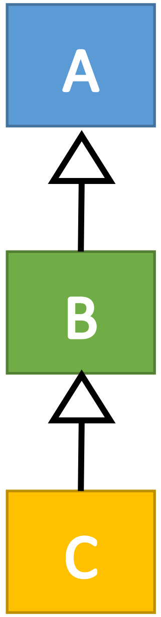
Inheritance (is-a) Downcasting Pointers



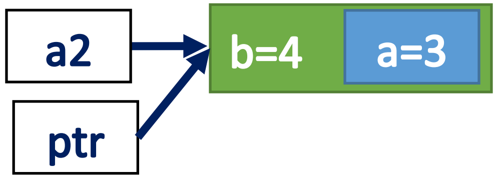


```
void main(){
   A * a1 = new A(2); //A’s pointer to A’s object
   a1->print(); //A’s print called.

   B *ptr = dynamic_cast<B*>(a1);
   if (ptr != NULL)  //return null when failed
    ptr->print();

   // Type Casting failed as A’s pointer is pointing
   to A’s object
   // Through Null check we can avoid run time error


}
```

Inheritance (is-a) Downcasting Pointers

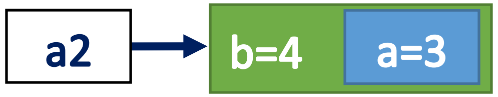


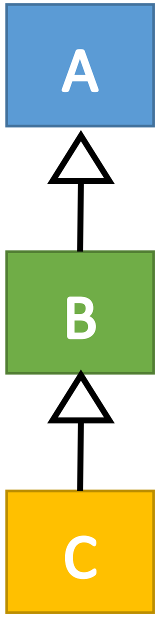
```
void main(){
   A * a2 = new B(3, 4); //A’s pointer to B’s object
   a2->print(); //B’s print called.

   B *ptr = dynamic_cast<B*>(a2);
   if (ptr != NULL)  //return null when failed
    ptr->print();

   // Type Casting is successful because A’s pointer is pointing to B’s object
   // Not create new object just perform down casting of same object for derived class pointer


}
```



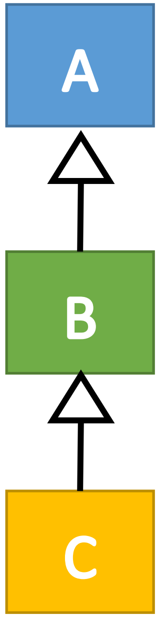
Inheritance (is-a) Downcasting Pointers

```
void main(){
   A * a2 = new B(3, 4); //A’s pointer to B’s object
   a2->print(); //B’s print called.

   C *ptr = dynamic_cast<C*>(a2);
   if (ptr != NULL)  //return null when failed
    ptr->print();

   // Type Casting failed as A’s pointer is pointing
   to B’s object
   // Through Null check we can avoid run time error


}
```

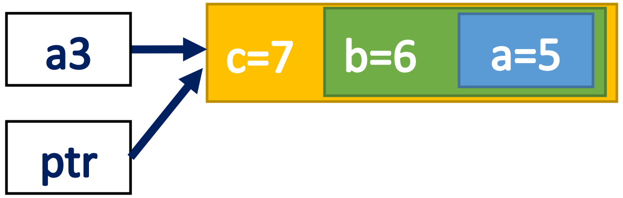




Inheritance (is-a) Downcasting Pointers



```
void main(){
   A * a3 = new C(5, 6, 7); //A’s pointer to C’s
   object
   a3->print(); //C’s print called.

   C *ptr = dynamic_cast<C*>(a3);
   if (ptr != NULL)  //return null when failed
    ptr->print();

   // Type Casting is successful because A’s pointer
   is pointing to C’s object
   // Not create new object just perform down casting
   of same object for derived class pointer

}
```

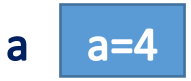


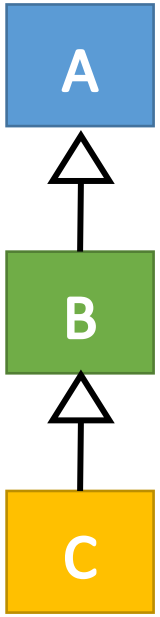
Inheritance (is-a) Down casting References

- Down casting converts base class reference to derived class object,
  - if base class is pointing to derived class object.
  - `dynamic_cast` operator is used for down casting References
    - Determine object's type at runtime
    - No way to check, if not of proper type (cannot be cast)
    - Exception is generated by system for bad cast error.
  - `dynamic_cast` will not work
    - With protected and private inheritance
    - With classes, which not have any virtual functions.
- Down casting is helpful
  - For accessing explicitly derived class data and functions that does not exist in base class.

Inheritance (is-a) Down casting References

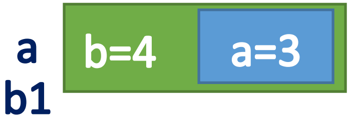
```
void main(){
   A & a = A(4);
   a.print(); //A’s print called.

   try{
      B & b1 = dynamic_cast<B &> (a);
      b1.print();
   }
      catch (bad_cast e){ //throws bad cast error.
      cout << e.what()<<endl;
   }
   // Type Casting failed as A’s reference is to A’s
   object
   // Bad Cast Error is generated by system


}
```

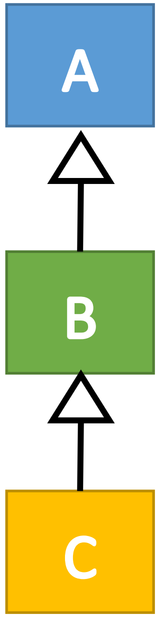




Inheritance (is-a) Down casting References

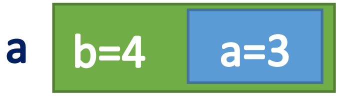
```
void main(){
   A & a = B(3, 4);
   a.print(); //B’s print called.

   try{
      B & b1 = dynamic_cast<B &> (a);
      b1.print();
   }
      catch (bad_cast e){ //throws bad cast error.
      cout << e.what()<<endl;
   }
   // Type Casting is successful because A’s reference
   to B’s object
   // Creates new object by calling copy constructor
}
```

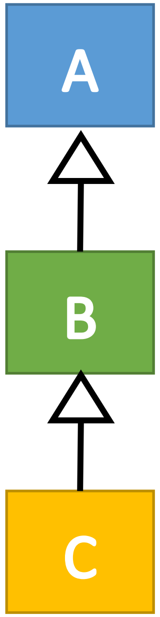




Inheritance (is-a) Down casting References

```
void main(){
   A & a = B(3, 4);
   a.print(); //B’s print called.

   try{
      C & c1 = dynamic_cast<C &> (a);
      c1.print();
   }
      catch (bad_cast e){ //throws bad cast error.
      cout << e.what()<<endl;
   }
   // Type Casting failed as A’s reference to B’s
   object
   // Bad Cast Error is generated by system
}
```





Inheritance (is-a) Down casting References

```
void main(){
   A & a = C(5,6,7);
   a.print(); //B's print called.

   try{
      C & c1 = dynamic_cast<C &> (a);
      c1.print();
   }
      catch (bad_cast e){ //throws bad cast error.
      cout << e.what()<<endl;
   }
   // Type Casting is successful because A's reference
   to C's object
   // Not Create new object
}
```