333 Section 6 - C++ Inheritance

Welcome back to section! We're glad that you're here :)

C++ Inheritance

Access Specifiers:

- `public:` visible to all other classes
- `protected:` visible to this class and its *derived* classes
- `private:` visible only to the current class

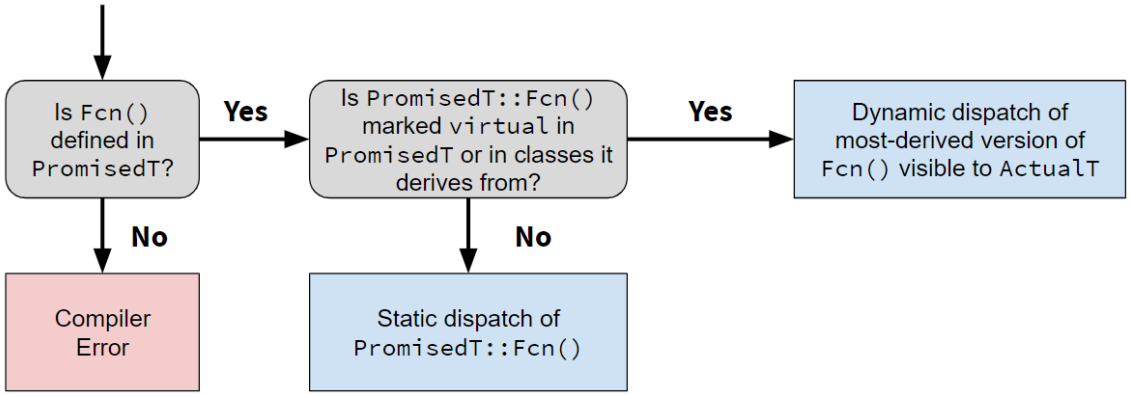
What's different in C++ (compared to Java)?

- *Static vs. dynamic dispatch* – in Java, all method calls are dynamic dispatch
- Pure virtual functions, abstract classes, why no Java "interfaces"
- Assignment slicing, using class hierarchies with STL

`virtual` keyword: Prefix a member function's declaration with this to use dynamic dispatch. Note: derived (child) functions don't need to repeat the virtual keyword, but it traditionally often do.

`override` keyword (C++11): Postfix a member function's declaration with this to tell the compiler that this method should be overriding an inherited virtual function – good to use if available.

```
PromisedT *ptr = new ActualT();
ptr->Fcn();  // which version is called?
```



Exercise:

1) Inheritance & Virtual Function

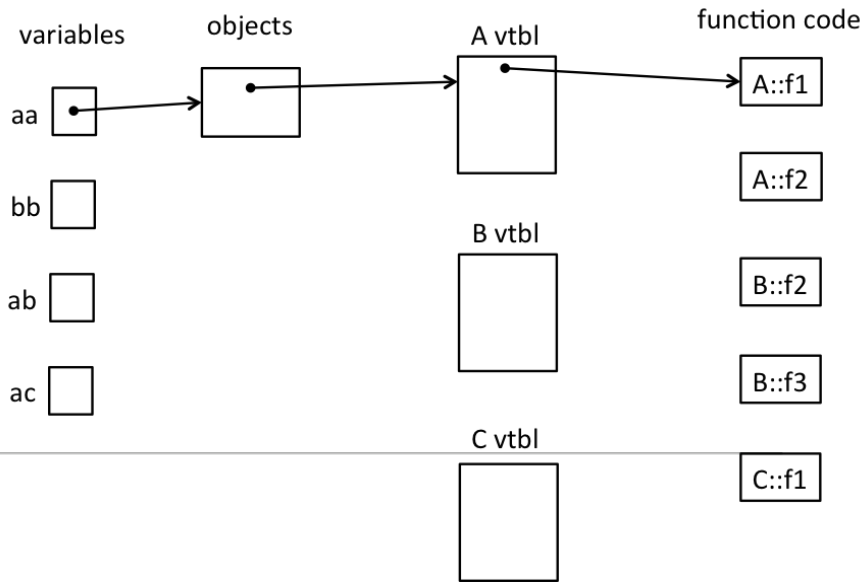
Consider the program below, which does compile and execute with no errors, except that it leaks memory (which doesn't matter for this question).

(a) Complete the diagram on the next page by adding the remaining objects and all of the additional pointers needed to link variables, objects, virtual function tables, and function bodies. Be sure that the order of pointers in the virtual function tables is clear (i.e., which one is first, then next, etc.). One of the objects and a couple of the pointers are already included to help you get started.

(b) Write the output produced when this program is executed. If the output doesn't fit in one column in the space provided, write multiple vertical columns showing the output going from top to bottom, then successive columns to the right

```
#include <iostream>
using namespace std;

class A {
 public:
  virtual void f1() { f2(); cout << "A::f1" << endl; }
  void f2() { cout << "A::f2" << endl; }
};
class B : public A {
 public:
  virtual void f3() { f1(); cout << "B::f3" << endl; }
  virtual void f2() { cout << "B::f2" << endl; }
};
class C : public B {
 public:
  void f1() { f2(); cout << "C::f1" << endl; }
};
```




```
int main() {
  A* aa = new A();
  B* bb = new B();
  A* ab = bb;
  A* ac = new C();
  aa->f1();
  cout << "---" << endl;
  bb->f1();
  cout << "---" << endl;
  bb->f2();
  cout << "---" << endl;
  ab->f2();
  cout << "---" << endl;
  bb->f3();
  cout << "---" << endl;
  ac->f1();
  return EXIT_SUCCESS;
}
```

Output:

Ex2. Virtual holidays! Consider the following C++ program, which does compile and execute successfully.

```
#include <iostream>
using namespace std;

class One
{ public:
        void m1() { cout << "H"; }
virtual void m2() { cout << "l"; }
virtual void m3() { cout << "p"; }
};

class Two: public One {
public:
  virtual void m1() { cout << "a"; }
         void m2() { cout << "d"; }
  virtual void m3() { cout << "y"; }
        void m4() { cout << "p";}
};

class Three: public Two
{ public:
         void m1() { cout << "o"; }
         void m2() { cout << "i"; }
         void m3() { cout << "s"; }
         void m4() { cout << "!"; }
   };
```

```
int main() {
    Two t;
    Three th;
    One *op = &t;
    Two *tp = &th;
    Three *thp = &th;

    op->m1();
    tp->m1();
    op->m3();
    op->m3();
    tp->m3();

    op->m1();
    thp->m1();
    op->m2();
    thp->m2();
    tp->m2();
    tp->m1();
    tp->m3();
    thp->m3();
    tp->m4(); cout <<
    endl;
};
```

(a) (8 points) On the next page, complete the diagram showing all of the variables, objects, virtual method tables (vtables) and functions in this program. Parts of the diagram are supplied for you. **Do not remove** this page from the exam.

(b) (6 points) What does this program print when it executes?

(c) (6 points) Modify the above program by removing and/or adding the `virtual` keyword in appropriate place(s) so that the modified program prints `HappyHolidays!` (including the `!` at the end). Draw a line through the `virtual` keyword where it should be deleted and write in `virtual` where it needs to be added. Do not make any other changes to the program. Any correct solution will receive full credit.

(cont.) Draw your answer to part (a) here. Complete the vtable diagram below. Draw arrows to show pointers from variables to objects, from objects to vtables, and from vtable slots to functions. Note that there may be more slots provided in the blank vtables than you actually need. Leave any unused slots blank.