1 Basic multiple inheritance issues

http://www.adp-gmbh.ch/cpp/virt_base.html

```
class B
{
  protected: int b_;
};

class D1 : public B { };
class D2 : public B { };

class D1v : public virtual B { };
class D2v : public virtual B { };

class D : public D1, public D2
{
  public: void f();
};

class Dvv : public virtual D1, public virtual D2
{
  public: void f();
};

class D_vv : public D1v, public D2v
{
  public: void f();
};

void D::f()
{
  // b_= 4;
  // error C2385 D::b_ is ambiguous,
  // could be the 'b_' in base 'B' of base 'D1' of class 'D'
  // or the 'b_' in base 'B' of base 'D2' of class 'D'

  D1::b_ = 5;  // Ok
  D2::b_ = 6;  // Ok
}

void Dvv::f()
{
  // b_ = 4;
  // error C2385: 'Dvv::b_' is ambiguous
  // could be the 'b_' in base 'B' of base 'D1' of class 'Dvv'
  // or the 'b_' in base 'B' of base 'D2' of class 'Dvv'
```

```
  D1::b_ = 7;  // Ok
  D2::b_ = 8;  // Ok
}
```

Here is the implementation of `D_vv::f`. `D_vv` has to base classes: `D1v` and `D2v`, which both have `B` as a virtual base class. Therefore, `b_` is shared, and setting `b_` is equivalent to setting `D1v::b_` which is equivalent to setting `D2v::b_`.

```
void D_vv::f()
{
  b_=4; // Ok
  D1v::b_=5;
  D2v::b_=6;

  if (b_ != D1v::b_ ||
      b_ != D2v::b_ ||
  D1v::b_ != D2v::b_) {
throw "Should not be thrown";
  }
}

int main()
{
  D d;
  d.f();
  Dvv dvv;
  dvv.f();
  D_vv d_vv;
  d_vv.f();
  return 0;
}
```

2 Java interface vs. abstract class

http://java.sun.com/docs/books/tutorial/java/interpack/interfaceDef.html

- Interfaces cannot implement any methods. Can only declare methods and define constants.
- Classes can implement many interfaces. Unrelated classes can implement the same interface.
- Interfaces can have multiple super-interfaces. Can only have one superclass. Implementations of interface methods must have the same signature as the declaration.
- Interfaces can be used as types. From the URL above:

```
public class StockMonitor {
    public void watchStock(StockWatcher watcher,
                           String tickerSymbol, double delta) {
    ...
    }
}
```