Week 7 Discussion:

Among object-oriented languages, one feature that varies considerably is whether the language allows multiple inheritance. C++ does but Ada does not. Java takes a middle ground approach of allowing multiple inheritance of interfaces but not classes. Using a C++ example, illustrate some of the complexities that multiple inheritance introduces. How does C++ deal with them? Why does Java's middle ground approach offer some of the benefits of multiple inheritance while avoids its problems.

Darn I wanted to use a Car example because it fits so well, but I see Homer Thompson beat me to it. I guess I’ll use GlassBeaker:

//describing requirements of scientific beaker

Class GlassBeaker{

Public:

Virtual void glass();

};

//describing a small beaker

Class BeakSize1: public GlassBeaker {

Public:

Virtual void size();

Virtual void storageLocationA(); //Stored in closet A

};

//describing a reinforced beaker (handles hotter liquids)

Class ReinforcedBeaker: public GlassBeaker{

Public:

Virtual void reinforcedGlass();

Virtual void storageLocationB(); //Stored in closet B

};

// a beaker of small size that is reinforced

Class SmallReinforcedBeaker: public BeakSize1, public ReinforcedBeaker {

Public:

Virtual void storagelocation();

};

int main()

{

SmallReinforcedBeaker testingClass;

testingClass.num=3;

    return 0;

}

This is ambiguous you need to specify a specific storage location (which is how C++ deals with it).

SmallReinforcedBeaker. Storagelocationb::num=3

Java avoids Inheritance issues by using Interfaces, all of which must be overridden in a subclass.