Discussion 7

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.

The below simple program illustrates how C++ deals with a multiple inheritance:

class firstName

{

Public:

void firstName() { cout << "Your First Name"; }

};

class lastName

{

Public:

void firstName() { cout << "Your First Name"; }

};

class fullName : firstName, lastName { };

int main()

{

fullName n;

n();

return 0;

}

Multiple inheritance has increased complexity and ambiguity in situations such as the "diamond problem", where it may be ambiguous as to which parent class a particular feature is inherited from if more than one parent class implements that feature. C++ deals with the complexities of multiple inheritance by using the “virtual” function which only allows the constructor to be called for that inheritance once at runtime. Java prevents this from happening by not allowing the multiple inheritance.