Multiple Inheritance, Abstract Classes, Interfaces

Written by John Bell for CS 342, Fall 2018

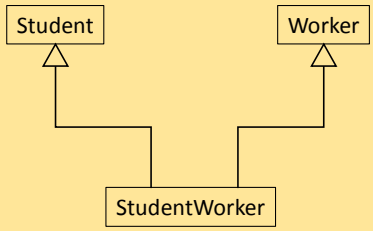
Based on chapter 8 of "The Object-Oriented Thought Process" by Matt Weisfeld, and other sources.

Frameworks

- Frameworks provide structure and support suitable for use in a variety of (related) programs.
- For example, most Microsoft Office programs have similar toolbars, menus, key commands and general look-and-feel and behaviors.
- The Java API includes GUI classes suitable for many apps, such as windows, buttons, menus, dialogs, …
- Specialty frameworks may apply to a particular domain, such as e-commerce. (shopping carts, ...)
  - JUnit is a well-known framework for conducting (automated) unit tests of Java code. See also xUnit.

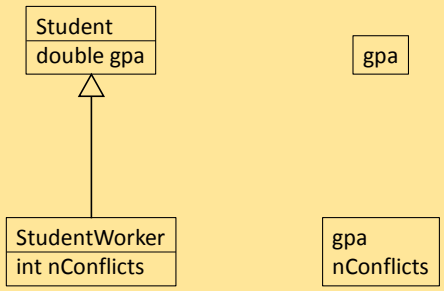
Multiple Inheritance

- Languages such as C++ allow classes to inherit from multiple sources, which can be quite useful:



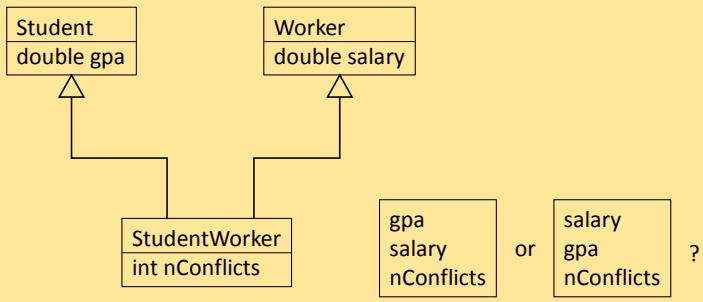
Storage of Inherited Fields

0. In a descendant object, inherited data is stored first, followed by newly defined data fields.



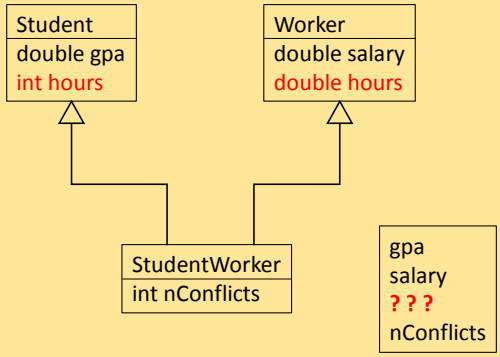
Multiple Inheritance Problems

1. How to store inherited fields



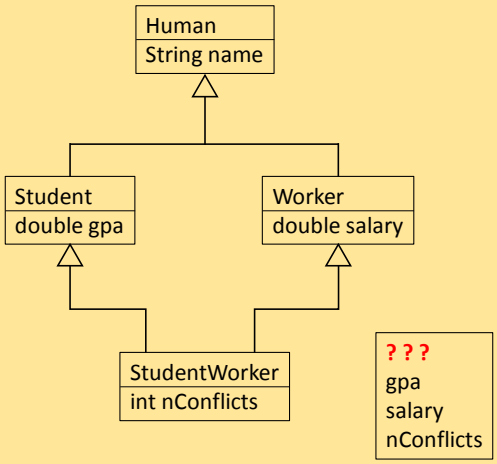
Multiple Inheritance Problems

2. How to store and resolve multiply inherited fields



Multiple Inheritance Problems

3. How to store inherited fields from a common ancestor (The “diamond problem”.)



Abstract Methods form a “Contract”

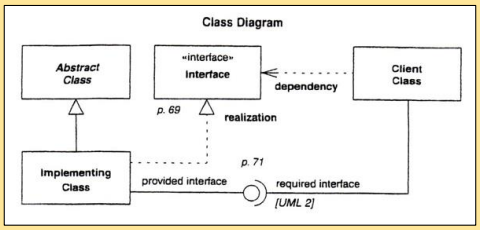
- Abstract methods are declared as required components of a class, but not provided. This forms an obligation to provide these methods, in the form of a “contract”.
- Abstract classes cannot be instantiated.
- Descendants of abstract classes must either fulfill the obligations of the contracts declared in all of their ancestors, or else they are also abstract.
- A class with ONLY abstract methods is termed “pure abstract”.

Inheriting from Abstract Classes

- Inheriting from abstract classes can alleviate (some of) the problems of multiple inheritance.
  - Since the methods were not provided in ancestor classes, they are not inherited in descendant classes, alleviating some of the storage and resolution problems.
  - Providing the required methods can satisfy the contractual obligations of multiple ancestors with a single method, provided there are no conflicts in argument signatures. (Overriding, not overloading.) This also satisfies the same obligation inherited through multiple ancestor paths (the diamond problem.)

Interfaces

- Interfaces are like pure abstract classes - They declare contractual obligations for implementers without providing any implementations.
- "UML Distilled" by Fowler diagrams interfaces as:



- (StarUML does not diagram interfaces well. ☹)

Benefits of Interfaces

- Interfaces define common required properties (methods) that different classes must provide, without incurring the full overhead of inheritance.
- Classes implementing a common interface are not related by either parent-child or sibling relationships. They merely share some common characteristics.
- Interfaces support polymorphism, with a common set of method signatures for all implementing classes.

Interfaces in Java

- Java does not allow multiple inheritance, to avoid all the problems discussed earlier.
- However a Java class may implement as many Interfaces as it wishes.
- Since there are now no actual methods or fields inherited, there are no conflicts.
- An interface can extend another interface (but only 1).
- All Interface methods are public and abstract. And all the fields are public, static, and final.
- Interfaces can't be instantiated, but variables of an Interface type can refer to any implementing class.
  - (someObject instanceof someInterface) will return true if someObject implements someInterface.

New in Java 8

- Java 8 now allows Interfaces to define default methods, only available when no other is provided.
  - Overriding a default method makes the default unavailable. (default not reachable with “super”.)
  - In the event of conflicts between multiply “inherited” default methods, all default methods become null and void. In this case the class is required to provide the method(s), just as if there had not been any defaults to begin with.
- Java 8 also now allows static methods in Interfaces, callable only through the Interface name, and not inherited.

A few common Java Interfaces

- Throwable - Any class that implements the Throwable Interface can be thrown from a try{ } block and caught in a catch( ) { } block. It is known that anything caught in a catch block must implement the Throwable interface.
- Runnable - The Java Runnable Interface specifies the run( ) method, required for any class to be run in a thread.
- Serializable - Implemented by classes whose state can be transmitted in a serial stream of data.

Review

Which of the following is NOT a problem with multiple inheritance?

A. How to store fields inherited from multiple parents, even when the fields are different.
B. How to resolve conflicts between the same field name inherited from different parents, especially when the data types are different.
C. How to handle the same field inherited from a common ancestor via multiple parents (The diamond problem.)
D. None of the above. That is to say that all of the above are problems.
E. The sparkling ruby problem.