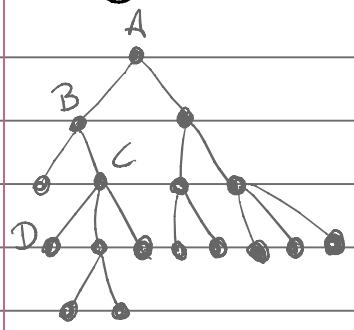
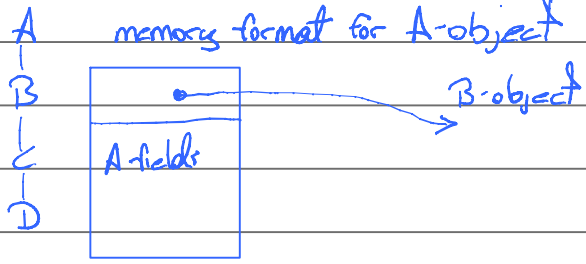
316 11-17-09

Note Title 11/17/2009

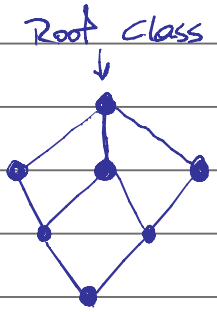
In single inheritance trees, there's a unique path from any ancestor class to any descendant class.



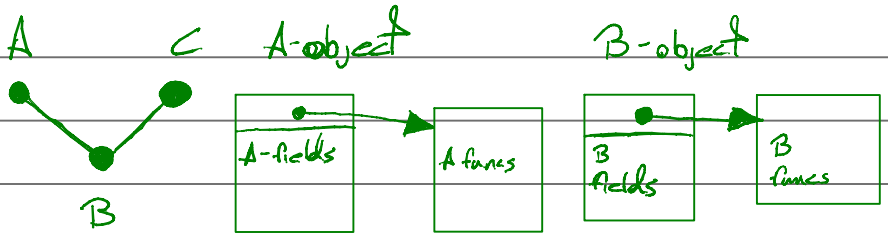
Access to class fields / functions can always be done uniformly by constant offsets fixed at compile time



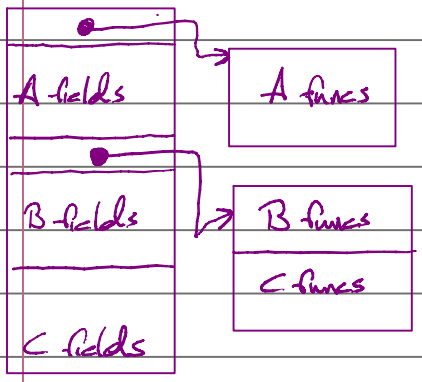
Multiple Inheritance hierarchy takes the form of lattice



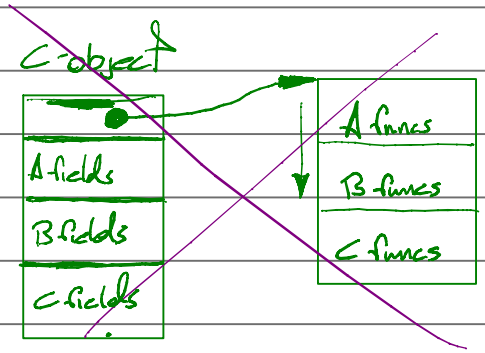
May be multiple paths from an ancestor class to a descendant class.



One possible mapping for C-object



2-function tables



A a;

a may reference A-object or C-object

No problem.

C c;

c may only reference C-object

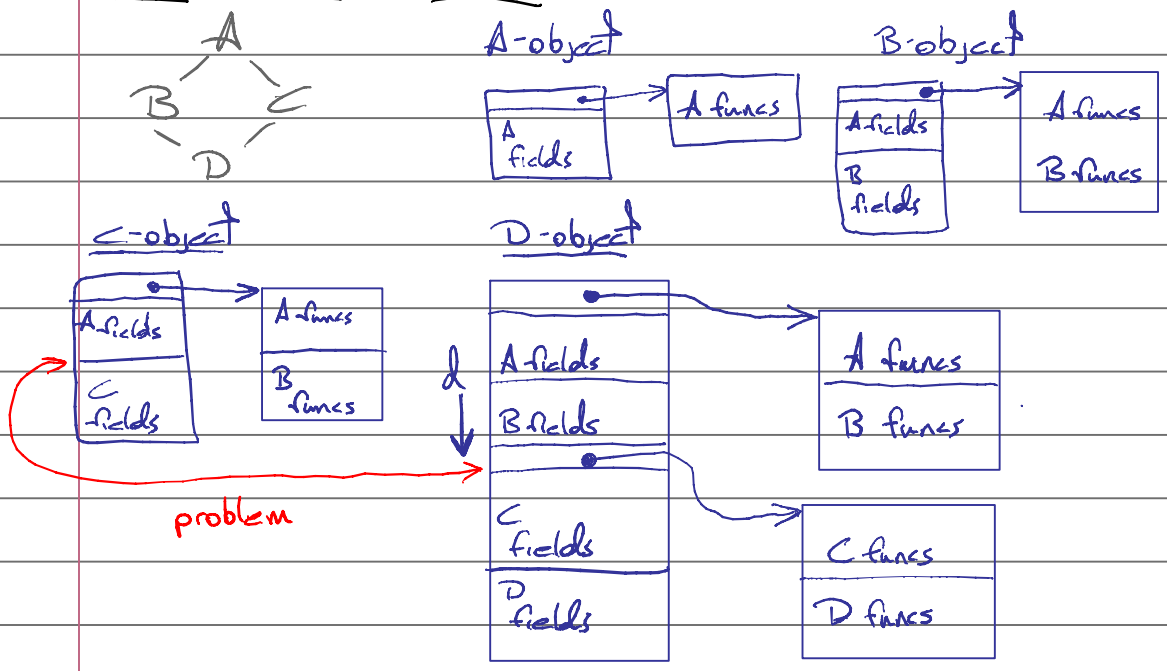
No problem.

B b;

b may reference B-object or C-object

- If b references B-object, offset = 0 is dynamically added to access B-fields/functions
- If b references C-object, offset = d is dynamically added to access B-fields/functions

Diamond Multiple Inheritance



C c;

c may reference C-object or D-object

IF c references C-object, offset = 0 is dynamically added to access C fields/funcs

If c references D-object, " = d " " " " " D fields/funcs

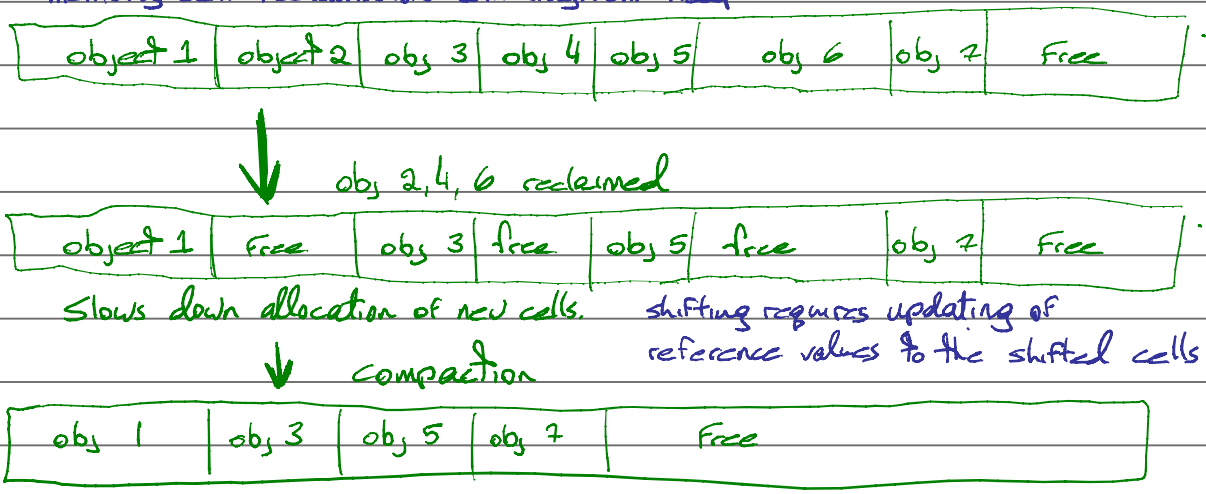
Heap Memory Management

Heap = memory region where dynamically created data/objects pointed to by references/pointers are allocated

- class objects in OOPL's
- data objects pointed to by "pointers" in C, ADA
- data objects used in LISP and PROLOG

Tasks of Heap Memory Manager

- Dynamic allocation of cells
  - constructor functions used in conventional languages like C/C++, ADA, Java, Eiffel, etc.
  - LISP, PROLOG, Javascript, etc. have no constructor functions. Necessary cells are automatically allocated
- Reclamation of cells (deallocation)
  - destructor functions (delete(), free(), dispose(), etc) in C/C++, ADA, etc.
  - automatic garbage collector: Java, Eiffel, SmallTalk, LISP/PROLOG
- Compaction (Defragmentation)
  - memory cell reclamation will fragment heap



Common Mapping of Entire Main Memory Segment Allocated to User Programs