Advantages of OO Design / Programming:

In general:
- Code reuse: Less defects in code when reused since it’s exercised in many applications.
- Data hiding: Object data and algorithms are hidden from user to simply use via published API.
- Extensible: Write code today that solves both today’s problems as well as future problems with minimal or no code changes.
- Layering/Modules: Location to identify defects is narrowed due to understanding of the modules and layers.
- Natural: Designing systems can be done via mimicking object and their interactions in the real world.
- Division of Labor: Different engineers can work on different objects such that solution is built in parallel.

C: w/ void pointers and function pointers:
- Most portable: oldest language with the most compilers for the most processors.
- Lightest/Fastest: Least language complexity such that the compiled code would be the simplest at compile time.
- Destructor removed authority and access: Less chance of undetected errors by avoiding unauthorized access to memory.
- Smallest executable: Most suitable for use in embedded systems with limited memory.
- Least magic: all code to execute was present and expected.
- Logic decisions based on existence of virtual function: More design choices for the software engineer.
- Constraint methods: easily identifiable from the parameter to new_List/new_Stack/List object

Base/Class Derived Class (extending Base in Java):
- Rich language Features: more design choices for engineer (also benefit of C++)
- Garbage collector: less focus was on memory management than with C or C++
- Decisions are made for you simplifying choices:
  o Object / arrays are all heaps
  o Only derivation is public
  o Only use of ‘dots’ to access data members
- Rich set of APIs and a large user community: code for many tasks already exists many are free
- .jar can are platform independent, larger possibilities for installations without extra effort
- More compile time checking (object in container must be derived from Base): more errors identified at compile time (also C++)
- Constraint methods: easily identifiable from the Base class definition

Base/Class Derived Class (:public Base in C++):
- Overloaded operators: allowed to treat user defined objects with the same syntax as primitives.
- Const keyword: compile can enforce more design choices, gave programmer more information about code.
- References: power of pointers without pointer syntax, ensure no null pointer exception
- Inline expansion (defined method in class definition): organization of a function call without function call overhead.

Java w/ Generics:
- Homogeneous container: more compile time checking of our design (also C++ templates)
- Smaller executable than with templates:
  - One set of code is created for all uses of Generic code

C++ w/ templates
- Refer to benefits of templates from prior lecture

C++ w/ diskfiles
- Persistence: data existed longer than program execution, more like most applications
- Less memory used: all TNodes were only in memory as needed.
- More object could be stores: Limited by disk space rather than memory.
- No chance of memory leaks: all TNodes were allocated on the RTS.
- Flat data: objects could be assigned, written and read to disk as a unit rather than field by field
- Repeatability in testing – reading an ASCII test file could continue rather than ending

Disadvantages:

In General:
- High learning curve: sophisticated solution that required more effort to understand and implement.
- Overkill: Simple problems can be solved without as much OO infrastructure.
- Debugging generic code is a challenge: Takes more initial time to get solution produced.
- Much magic: It’s not always obvious upon inspection which code is getting executed.

C:
- Track your own memory: Engineer needs to focus on the problem to solve and use of memory at the same time.
- Lack of OO syntax: OO ideals needed to be implemented from scratch rather than with provided language features.
- Most syntactically complex: Mixture of *, ->, .., () in working with virtual methods/function pointers.
- Too many pointers: A large amount of code is checking the pointer to not be null rather than logic.
- Least compile time checking: All items in container were void pointers so more run-time errors than compile errors.

Java/C++ with Base class/Derived class:
- Not natural: Objects stored in the container needed to be derived from an abstract parent.
- Not natural: Parameters to methods needed to match abstract parent rather than being of type of object.
- Polymorphism was limited: No possibility for virtual constructor or virtual friend.
- Container was heterogeneous – homogeneous design not enforced by the compiler.
Java w/ Generics:
- Generics is compile-time only operation: still had to use virtual table and function pointers (behind the scenes) for polymorphism.
- Object still had to extend base: Although type of object being stored is known, we still have to treat object using abstract parent.

C++ w/ Templates:
- Compiler checks were more verbose than necessary- more straight forward explanation is possible.
- Extra syntax (template class <Whatever> prefix) to every block – maybe present at the file level could be sufficient
- Largest executables – one set of object code for each use of a template container
- Constraint methods were not obvious – Method required to be implemented from the object being inserted weren’t stated
- Least portable solution – Same code won’t necessarily compile with all compilers

C++ w/ template and diskfiles
- Not best for security – deleted information remains are fragments in the datafile.
- Wasted space in the diskfile – TNodes exist on the disk that are no longer in the Tree.
- Debugging was difficult – had to rely on reading octal dumps to resolve problems.
- May be too slow – disk is slower than memory
- Most code – each line of memory code as expanded to two lines of disk access code.
- Highest complexity of memory managements – you had to track whether or not a TNode was synchronized to disk or could be discarded.
- Solution requires external storage – imbedded devices may not have such storage