OOP Recap tutorial
combined with questions from past exams

236703 Object-oriented programming ○ CS, Technion
Natan Bagrov ○ January 2018 (minor changes, January 2019)
What’s for today

- Basics
  - Modifiers
  - Methods’ stuff
- Java
  - Inheritance
  - Java 1.8 features
- C++
  - Object structure
  - Multiple & virtual inheritance
  - Casts
- Squeak
  - Basics
  - Object model
- Generics
  - In C++, C# and Java
- Your requests: selected exam questions
Basics: modifiers

- What static modifier means? Can we override static methods? Do we need an instance to access static field/method?
- What final modifier means? Can we override final methods?
- What virtual modifier means? How can we achieve such behavior in Java?
Basics: methods’ stuff

- Method overloading - what is it?
- Method overriding - what is it?
- Method hiding - what is it?
- Static binding - what is it? What kinds of methods are statically bound in C++, Java, Squeak, C#?
- Dynamic binding - what is it? What kinds of methods are dynamically bound in C++, Java, Squeak, C#?
- ‘MultiMethods’ - what is it?
Java: interfaces

- May extend any number of interfaces
- May contain public methods (implicitly public)
- May contain public static final fields
- May contain methods with implementation. Such methods must be declared with default modifier - available since Java 1.8
- Interfaces containing one non-default method can be used as functional interfaces - remember what that is? When doing that, it is better to mark this interface as @FunctionalInterface - why?
Java: interfaces - default methods

- May introduce ambiguity - what that is? Can we get ambiguity with Java 1.7?
- Can we have coincidental ambiguity? Inherent? - what these ambiguities are?
- How can we ‘fix’ such ambiguities?
Lambda expressions are instances of an anonymous classes that implement (the anonymous classes) a functional interface.

- Lambdas are ‘first-class-citizens’ - what that is?
- Lambdas have closure - what that is?
Java: streams

- A stream is a sequence of elements that support sequential and parallel processing.
- Streams let us elaborate regular methods to serve as a ‘higher-level’ methods, producing a much cleaner code.
- There are 2 kinds of operations on streams: intermediate and terminal - what each operation produces (return value)? What is the main difference between these 2 operations?
- Every stream is a ‘one-timer’ - what that is?
- Unlike regular sequences (collections), streams can be infinite.
Pop quiz

Q2:


Q3:

C++: object structure & multiple inheritance

- What is multiple inheritance? Does Java, C# or Squeak support it?
- Java and Squeak have the `super` keyword, C# has the `base` keyword, why don’t C++ have something like it?
Virtual pointers and virtual tables are not part of the C++ specifications, so why do we need vptrs and vtbls?
Where is the vptr located (gnu, borland)? What is the advantage of each solution over the other?
Are vptrs and vtbls used inside constructors?
Why do we need to have ‘as-if-static’ behavior inside a constructor?
During an invocation of a method, `this` may be adjusted - why do we need to adjust `this`?

What is thunk?

Multiple inheritance may introduce ambiguity. There are 2 kinds of ambiguities (inherent and coincidental) where the code compiles unless we use the ambiguous method, and a special case of inherent ambiguity where the code did not even compile. Can you demonstrate?

How can we ‘fix’ such ambiguities?
C++: virtual bases

- What is a virtual inheritance?
- Where is the virtual base located in the memory layout of the object?
- What class is responsible of instantiating such virtual bases?
C++: initialization order

1. Apply topological sort, and rank
2. Construct all virtual base classes, use ranking order
   ○ If a virtual base class derives from a non-virtual class, instantiate parent-class first
3. Construct all non-virtual base classes, use ranking order
C++: casts

- What is upcast? What is downcast?
- Why can't we use `static_cast` to downcast from a virtual base? Can we use it for upcasting?
- Why can't we use `dynamic_cast` on non-polymorphic class?
- What is `reinterpret_cast`?
Pop quiz 2

Q3:


Q1:


Squeak: blocks

- What blocks are?
- Blocks are also ‘first-class-citizens’
- Blocks execute in the context in which they were defined and may refer to independent variables (simply: they ‘remember’ their environment)
- What does a block return? Where is it returned to?
- What is a non-local return? Do we need to be cautious with it?
Squeak: object model

- Everything is an object
- Every object is an instance of one class
- Every class has a single superclass
- Everything happens by sending messages to objects
- Method lookup follows the inheritance chain - how is the lookup done?
Pop quiz 3

Q2:


C++: templates | Java, C#: generics

- Why can’t we have a template virtual methods, in C++?
- Can we have template classes with virtual methods?
- Say a template/generic class was used with n different types, how many classes will be generated during compile-time / run-time in C++? How many in Java? How many in C#?
- What is type-erasure in Java? Why can’t we use primitives (like int) as generic parameters in Java?
- Why does C# more informative at run-time than Java when it comes to generics?
- We said that methods in Java are virtual by default; we said that in C++ a method can’t be template virtual; why does it work in Java?
Conformance

- What does co-variance mean when it comes to overriding methods? Contra-variance? No-variance?
- Does it makes sense to override a method with a co-variant return type?
- Does it makes sense to override a method with a contra-variant parameter type?
Generics: conformance

- In Java and C#, if B derives from A, does `List<A> la = new ArrayList<B>()` legal?
- So, is this legal: `new List<A>().Add(new B())`?
- What does `<? extends A>` mean? What does `<? super A>` mean?
- What does `<T> where T : A` mean? What does `<T> where A : T` mean?
- Does these 2 bullets actually mean the same (in Java / C#)? What is the difference? So, what, in C#, means the same as the first bullet?
Pop quiz 4

Q3:


Q3d:


Your requests...
Your requests, sorted by date*


Looks like questions 1 and 3 are the most problematic, we'll take that into consideration ;)

*The date the email was sent. Note that this list updated to Jan 2018.
GOOD LUCK!

Welcome to our company. We use object-oriented methodologies here.

Thanks to that our code is always clear and simple!

Sample code:
Singleton::GetInstance().GetFactory().GetWrapper().GetProxy().GetAdapter
(Manager::GetInstance().GetObjectIndexByName(s)).GetAccessState().GetParam
(PARAM_ENUM_NAME).AsString(true)