Final exam – Object Oriented Programming (236703)

Spring 2013, Moed Bet

Summer 2013, Moed Alef

Lecturers: Spring semester – Dr. Keren Lenz
Summer semester – Prof. Yossi Gil

Teaching assistants: Spring semester – Eran Gilad, Eyal Moscovici, Amit Carmeli
Summer semester – Boris Sosin

Instructions:

1. The exam contains 5 equally graded questions. You must answer all of them.
2. In case you don’t know the answer to a question or a sub-question, you may write "I don’t know" and receive 20% of the points.
3. Unless specified otherwise, on questions in which you are required to provide the output of given code, the code might not compile or run to completion. On such cases, you should describe the reason for the compilation or run-time error.
4. Write to the point. Adding irrelevant information (even if correct) might cause a deduction of points.
5. Make sure your handwriting is readable. No cursive please.
6. Start every answer on a new page.
7. No additional material is allowed.
8. The exam contains 7 pages. Make sure your form includes all of them.
9. The exam's duration is 3 hours.

Good luck!
**Question 1: Squeak**

Consider the following expressions in Squeak:
- `x superclass class`
- `x class class`

**a. For each of the above expressions**, is there a value of `x` such that an exception will be thrown upon the evaluation of the expression? If so, give one such example. Otherwise, explain why there is no such `x`.

**b. For each of the above expressions**, is there a value of `x` such that the evaluation of the expression yields `x` itself? If so, give one such example. Otherwise, explain why there is no such `x`.

Recall that the method lookup algorithm taught in class looks for the method in the object's class. If the method is not found, the algorithm looks for the method in the parent class of that class, and so on and so forth. Consider the following code that implements the method lookup algorithm in Squeak:

```smalltalk
x := self class.
[x = nil] whileFalse: [
  [ x includesSelector: selector ] ifTrue: [
    "call the appropriate method and stop the search"
    x := ______________.
  ]
] "throw an exception (if method wasn't found and invoked)"
```

c. Complete the missing expression according to the method lookup algorithm taught in class.

d. Assume now that the missing expression is replaced with the expression `x class`. In each of the following cases, describe what the algorithm does, i.e. describe which classes does the algorithm traverse and what is the return value.

In all of the following cases, `A` is a class that inherits from `Object`. In the metaclass of `A`, the method `new` is overridden and returns an `Object` of type `A`, i.e. there is a definition for the method new in the meta-class of `A`.
- `(A new) superclass`
- `(A new) new`
- `(A new) isMeta
  (Note: isMeta is defined in Metaclass and returns true).

e. In C++, when a message is sent to an object, there is no need to execute a method lookup algorithm. This differs from Squeak's method invocation process. Explain why. Which property of the C++ language allows that?
Question 2: Multi-Methods

Recall that in C++, Java and C#, when a method is overloaded by a method with the same name, the choice of which method to execute is determined by the static types of the arguments, and not by their dynamic types.

a. Write a short program in any of the above languages that demonstrates this selection strategy.

We would like to add a feature to Java that will support dynamic choice of the method when overloaded, i.e. the choice between methods will occur during runtime according to the **dynamic type of the arguments** (this was described in the lectures as MultiMethods).

```java
void f(Base) {...}
void f(Derived) {...}

void g() { Base arg = new Derived();
    f(arg); }
```

The static type of `arg` is `Base`. Without supporting MultiMethods, the first `f` is executed. Our goal is to support choosing between the methods during runtime, so that the second `f` will be executed.

One can claim that such support in Multi-Methods can compensate for the limitations of Java generic's type erasure.

b. Explain how the added feature will improve Java generics.
   **Hint:** Consider how the compiler chooses the method to be executed, when the call for the overloaded method is executed from **within** a generic method.

In order to support MultiMethods, compile time and/or run time changes must be made.

c. Suggest a mechanism that will extend Java to support MultiMethods. Specify whether that mechanism operates at compile time, run time or both.

In a given program number of types is strictly bound, and that number is known at compile time.

d. For this program only, is there any way to implement MultiMethod support without incurring a significant run time overhead? If so, explain how. If not, explain why.

Currently, Java's compiler detects type errors in method invocations, hence such errors cannot exist at run time.

e. Supposing that the MultiMethod feature is supported, which run time error could happen on a program that has overloaded methods?
**Question 3: C++ Templates**

Consider the following main function in C++:

```cpp
int main() {
    C<A> ca1( A(5) );
    C<A> ca2( A(3) );
    C<B> cb1( B("hello") );
    C<B> cb2( B("world") );
    cout << "ca1.add(ca2) = " << ca1.add(ca2) << endl;
    cout << "cb1.add(cb2) = " << cb1.add(cb2) << endl;
}
```

In addition, consider the following classes:

```cpp
class A { … }; class B { … };
```

a. Implement the classes A, B and C such that:
   - The main function will compile successfully
   - The output of the main function will be:
     - `ca1.add(ca2) = 8`
     - `cb1.add(cb2) = hello world`

b. Assume that adding the following line to the above main function:

   ```cpp
   cout << "ca1.try(ca2) = " << ca1.try(ca2) << endl;
   ```

   will add the following line to the output:

   ```cpp
   ca1.try(ca2) = success
   ```

   However, adding the next line instead will cause a compilation error:

   ```cpp
   cout << " cb1.try(cb2) = " << cb1.try(cb2) << endl;
   ```

   Modify classes A, B and C so the above will hold.

c. Is it possible to write a class equivalent to C in Java? If so, re-write A, B and C in Java. Otherwise, explain why not.

(Question continued on the next page)
d. Consider a template class called Positive that is instantiated with an integer argument. If the argument is negative, a compilation error should occur. Fill in the numbered lines so that this requirement will hold.

```cpp
template <bool b> struct Guard {
    static void f() {}
};

template <> struct Guard<false> {
    //1
};

template <int n> struct Positive {
    static void check() {
        //2
    }
};

For example:
Positive<1>::check();       // compiles
Positive<0>::check();       // does not compile
```
**Question 4: Overriding Fields**

On the following question we shall consider extending C++ to support virtual fields. A virtual field is qualified by the keyword `virtual`, and is dynamically bound.

Consider the following classes:

```cpp
class A {
public:
    int field;
    A(int f) { field = f; }
    string toString() { return("field = " + field); }
};
class B {
public:
    int field;
    B(int f) : A(0) { field = f; }
};
```

a. What will the following code print?

```cpp
A* a = new B(3);
a->toString();
```

b. How would your answer to (a) if each one of the following changes is made (separately):
   1. Define the fields in A and B as `virtual int field`
   2. Define the field in B only as `virtual int field`

c. Can the type of an overridden field be changed in a covariant or contravariant way while maintaining compliance with the rules of conformance?

d. How would you implement the virtual fields' language extension? Describe the required data structures and algorithms.

e. One of the reasons why virtual fields are not supported on object oriented programs is the ability to obtain similar functionality by dynamic binding of functions. Modify the classes A and B so that the code below them will print the same value as printed in (b.1), but without supporting virtual fields. You may add additional fields and methods.

f. Are C# properties an approximation to virtual fields? Explain.
Question 5: The Object Model

a. Give an example of a language whose object model has 2 levels. Give an example of a language whose object model has 3 levels.

b. Which class exists on a 3-level language but not on a 2-level language? What does that class describe?

Students from the Technion Guangdong Institute of Technology have decided to define a new dynamic language whose object model has 1 level.

c. Where should methods be defined in such a language? How should methods be added? Can methods be overridden? Explain.

d. How can a new object be created?

e. How can it be determined whether two objects have the same type or not? Namely, what kind of type equivalence does the language use?

f. How can a "static" (shared among a few objects) variable be defined in the new language?