

THE UNIVERSITY OF HONG KONG

FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

CSIS0396A Programming Methodology and Object-Oriented Programming
(for Software Engineering and Double Degree students)

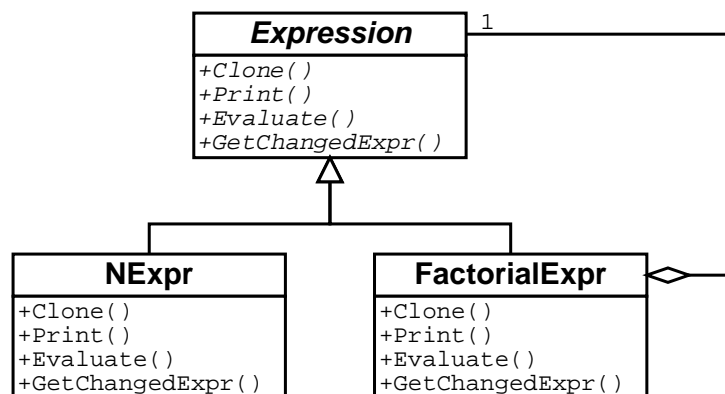
Date: May 16, 2002

Time: 9:30 am–12:30 pm

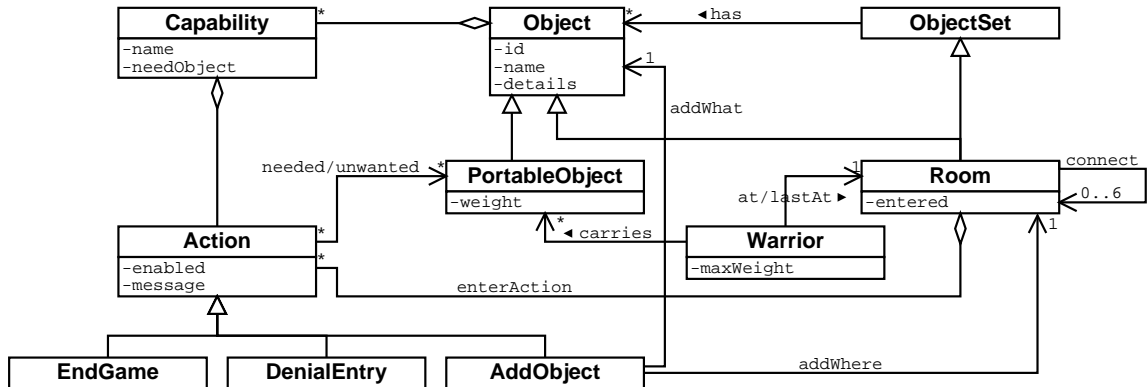
Candidates may use any calculator which fulfils the following criteria: (a) it should be self-contained, silent, battery-operated and pocket-sized; (b) it should have numeral-display facilities only and should be used only for the purpose of calculation; (c) it should not have any printing device, alphanumeric keyboard, or graphic display; and (d) it should not contain any recorded data or program. It is the candidate's responsibility to ensure that the calculator operates satisfactorily and the candidate must record the name and type of the calculator on the front page of the examination scripts. Lists of permitted/prohibited calculators will not be made available to candidates for reference, and the onus will be on the candidate to ensure that the calculator used will not be in violation of the criteria listed above.

Answer all questions in the answer book provided. Each of the 7 questions is worth 10%, and the marks of the 3 questions you score the most will be doubled.

1. (Compiling and linking)
 - a. (6%) There are two mechanisms to prevent multiple inclusions of the same code into the executable program: by the `#ifndef/#define/#endif` directives, and by the **extern** linkage specifier. Explain why we need two mechanisms rather than just one. Show examples of their usages.
 - b. (4%) What are the two ways of loading dynamic libraries? What are the steps needed for each of these usages? How to access symbols defined in the library in each of these usages?
2. (Copy-semantics) Consider the classes in the sequence guessing program of assignment 2. The class diagram of a few classes is shown below:



- a. (5%) The diagram shows that the expression class needs deep copy semantics. Why? Draw a diagram in which shallow copy semantics is used instead. How does this change the abstraction of an expression object?
 - b. (5%) Explain how to add copy-on-write to the class. Draw the new class diagram. You need a new class to keep track of the reference count of the *Expression* class. Write the copy constructor and the `GetChangedExpr` member function of the class.
3. (*Inheritance and Polymorphism*) Consider the Role-Playing Game program of assignment 3, with the following class diagram:



- a. (3%) Show the class definition of the `Action` class. You don't need to show the definitions of the member functions (only declaration of member functions are needed).
 - b. (3%) Suppose you want to add an action that adds a **new** capability to an object (rather than adding an object to a room in the `AddObject` action). What are the changes you need in the class diagram? Show the relation between the new class and the old ones. (You don't need to redraw the whole diagram. Just show the needed changes.)
 - c. (4%) Write the required code to implement part (b), by showing the class definition, the constructor which makes the object from an *istream*, and the member function that executes the action.
4. (*Multiple Inheritance*) In C++, a class can inherit from multiple classes. Consider the following classes:

```

class TA { ... };
class TB: public TA { ... };
class TC: public TA { ... };
class TD: public TB, public TC { ... };
class TE: virtual public TA { ... };
class TF: virtual public TA { ... };
class TG: public TE, virtual public TF { ... };           // Note the virtual!
    
```

- a. (5%) Suppose that each of the above classes has exactly one constructor. All such constructor takes the same integer as argument. Write the constructor of `TD`, and draw the object layout of an object of this type. Explain why it cannot be treated as an object of type `TA`.
- b. (5%) With the assumption in part (a), write the constructor of `TG`, and draw its object layout. Which part of the object layout will appear in all derived classes of `TG`?

5. (Generic Programming)

- a. (5%) Write a template function *UpCaseAll* which takes any STL container (excluding map) of strings as argument, and turns all lower-case characters of all strings contained to upper case. The function should require only that a forward iterator is defined.
- b. (3%) Suppose your program has a list of strings, and you want to use the above functions. Write the code to perform template instantiation for the vector and the template function, in case you want to perform explicit instantiation.
- c. (2%) When will you use explicit instantiation rather than implicit instantiation? Explain the benefit of explicit instantiation in this case.

6. (Exception)

- a. (6%) What is the output of the following code if you call *f(5)*? Explain your answer.

```
int g(int);
int f(int i) {
    try {
        for (; i < 10; ++i)
            cout << g(i);
    } catch(int n) {
        cout << n << endl;
    } catch(const char* c) {
        cout << c << endl;
    }
    return i;
}
int g(int i) {
    if (i >= 8)
        throw 2;
    if (i >= 7)
        throw "Hello";
    return f(i+2);
}
```

- b. (4%) With the help of an example, explain why exception handling code generated by the compiler needs to know whether a type is the derived type of another type at run-time. What information stored in the compiled program can be used for this?

7. (Scripting)

- a. (5%) Write a Python function *UpCaseDict* that takes one argument, which is a dictionary containing keys of string type mapping to other objects, and returns a new dictionary which is the same as the argument, except that all lower-case characters of all keys are converted to upper-case. (Hint: the *upper* function in the *string* module converts a string to uppercase and return it.)
- b. (5%) We say that Python is a reference-based language, with automatic garbage collection. With examples, explain what is meant by a reference-based language and what is automatic garbage collection, and compare it with a value-based language like C++.