

CS3723 HOMEWORK #8, Due: 2pm, 4/17/08

Note: If you collaborated with your classmates or used their notes, please note which classmates you collaborated with. If you use an external source, besides the text book, lectures, notes provided by the instructor, and your own intellect, please cite that source. Use quote marks if you are quoting material word-for-word from any source (including the text book).

DATA ABSTRACTION (Chapter 9)

1. (a) What is the oldest abstraction mechanism provided by programming languages? And what does it encapsulate?

Procedures, which encapsulate code.

- (b) List three features of data abstraction mechanisms provided by languages.

i. "identifying the interface of the data structure"

ii. "providing information hiding"

iii. "allowing the data structure to be used in different ways by different programs"

2. (a) What is the difference between an interface and a specification?

An interface is the externally visible parts (e.g., operation names, types, parameters) through which a client interacts with a component. There is often explicit language support for defining the interfaces and ensuring that the client and component implementation agree on it (e.g., public methods/fields in Java and declarations in header files in C). The specification is a description of the behavior of a component as visible through its interface. If the implementation does not satisfy the specification, then the client may not work correctly. But there is usually no automatic mechanism to determine if the implementation satisfies the specification. (This is one of the primary goals of unit testing.)

- (b) In ML's abstype construct only the abstract type name and the names and types of the values/functions declared in the abstype are externally visible. What is the externally visible interface of:

```
exception Empty;
abstype queue = Q of int list
  with
    fun mk_Queue() = Q(nil)
    and is_empty(Q(l)) = l=nil
    and add(x,Q(l)) = Q(l@[x])
    and first (Q(nil)) = raise Empty
      | first(Q(x::l)) = x
    and rest (Q(nil)) = raise Empty
      | rest (Q(x::l)) = Q(l)
    and length (Q(nil)) = 0
      | length (Q(x::l)) = 1 + length (Q(l))
end;
```

```

exception Empty
type queue
val mk_Queue = fn : unit -> queue
val is_empty = fn : queue -> bool
val add = fn : int * queue -> queue
val first = fn : queue -> int
val rest = fn : queue -> queue
val length = fn : queue -> int

```

OVERVIEW OF OBJECT-ORIENTATION (Chapter 10)

3. (a) List the four steps of Grady Booch's approach to object-oriented design.
 - i. *"Identify the objects at a given level of abstraction."*
 - ii. *"Identify the semantics (intended behavior) of these objects."*
 - iii. *"Identify the relationship among the objects."*
 - iv. *"Implement the objects."*
- (b) List the four main language concepts for object-oriented programming.
 - i. *dynamic lookup*
 - ii. *abstraction*
 - iii. *subtyping*
 - iv. *inheritance*
4. (a) What is dynamic lookup (a.k.a., dynamic dispatch)?

It is a mechanism that allows objects to respond in different ways to the same 'message' based on the dynamic identity of the receiving object.
- (b) How does dynamic lookup (e.g., a method invocation in Java) differ from an ordinary function call (e.g., in C)?

In an ordinary function call, the code that is being jumped to is determined by the name of the function (and perhaps overloading based on the type of the function's parameter) or by an explicit function value/pointer. With dynamic lookup, there is an implicit mechanism that determines the method implementation based on the dynamic identity of the receiving object.
- (c) What is the key word for declaring a member function with dynamic lookup in C++? *virtual*
5. (a) What do object-based abstraction and data abstraction both encapsulate (or as Mitchell says 'combine')?

functions (code) and data

- (b) What do both object-based and data abstraction ‘involve distinguishing’? *public interface and private data*
6. ML’s abstype feature allows one to create an abstraction that separates a hides (encapsulates) a single implementation behind its public interface.
- (a) How is this different from the public interfaces provided in object-oriented language (e.g., created by classes and interfaces in Java)?
In object-oriented languages multiple implementations (subclasses, implementing classes) can be plugged in to a client written to use a single public interface (that of the supertype).
- (b) Which two of the four object-oriented language concepts account for this difference between ML’s abstype and object-oriented classes?
subtyping and inheritance
- 7(a) What is the basic principle associated with subtyping? Give its name, and define it. *Principle of substitutivity: “If A is a subtype of B, then any expression of type A may be used without type error in any context that requires an expression of type B.”*
7. (b) How does Mitchell define inheritance? *The feature that allows new objects to be defined from existing ones.*
- (c) What is the difference between subtyping and inheritance? *Subtyping is a relationship among interfaces and describes when objects of a given class can be used. Inheritance is a mechanism for creating a class (the subclass) that reuses the implementation another class (the superclass).*
- (d) What keyword is used for both inheritance and subtyping in Java? *extends*
- (e) What is the keyword for the Java mechanism that provides only subtyping? *implements*
8. What is the difference between public, private, and protected members in Java? Define each one.
The public keyword is used to declare those parts of a class that make up its public interface which are visible to any client using objects of that class. The protected keyword is used to declare those parts of a class that are not part of the classes public interface, but nonetheless are part of the interface which is exposed to methods declared in subclasses so that they can interact with the inherited parts of the object. The private keyword is used to declare those parts of the class that are belong only to the internal implementation of the class and are not visible to clients of the class or to subclasses.

9. It is possible to mimic many aspects of objects in ML using records and closures. What is it about object-oriented programs cannot be mimicked using records and closures?

subtyping and inheritance

10. (a) What is a design pattern?

"A design pattern is a guideline or approach to solving a kind of problem that occurs in a number of specific forms."

- (b) List two design patterns.

The two described in the textbook are Singleton and Facade. In addition, the Visitor design pattern was described in lecture and in lab.

- (c) How does the organization of a typical object-oriented program differ from an equivalent program with a conventional function-oriented organization?

A program with a conventional function-oriented organization, will be divided up into functions to handle different logical operations. Each function may be further subdivided into separate cases for handling different types of data. The code related to each data type will be spread across several functions. In contrast, in a typical object-oriented program, the data type is the primary organization criteria and the operation is secondary to that. So the different code related to a single logical operation will be spread across the different classes in the class hierarchy.

- (d) What design pattern allows a program to use subtyping and dynamic lookup to achieve an organization/effect similar to the type case used on p. 288 in your text book? (The answer to this question is not in the text book.)

The Visitor Design Pattern describes a way of arranging objects so that one achieves a operation-oriented organization for code using object-oriented mechanisms (primarily subtyping and dynamic lookup).