

Problem 12.8

8. Explain the two problems with abstract data types that are ameliorated by inheritance.

The use of inheritance allows new data types to be created easily that are modifications of existing data types without affecting the original data type and, hence, legacy code that uses it. Furthermore, inheritance allows abstract data types to be hierarchical and share common elements if they have common ancestors. This makes code maintenance much cleaner and coherent because a change to a parent is inherited by all of the descendants.

Problem 12.9

9. Describe the categories of changes that a subclass can make to its parent class.

In general, subclasses can extend a parent class with few limitations by adding new data elements and methods. They can provide alternate implementations of methods that override methods in the parent class. They cannot make public elements (data/properties/methods) non-public since they must honor the public interface provided by the parent.

Problem 12.11

11. Explain the advantages and disadvantages of having all values in a language be objects.

If all values are objects then a much simpler, coherent programming environment is created because all objects/values are treated exactly the same. However, objects carry with them considerable processing/access overhead that is often too excessive given the amount of basic computation that is performed on simple value types in many programs.

Problem 12.12

12. What exactly does it mean for a subclass to have an is-a relationship with its parent class?

This means that a variable of the type of the subclass can appear anywhere in the code where a variable of the type of parent class can. This, of course, then means that a value of the type of the subclass can be assigned to a variable of the type of the parent class.

Problem 12.26

26. Can you define a reference variable for an abstract class? What use would such a variable have?

Yes, though you cannot instantiate an object of an abstract class. Reference variables of abstract classes can refer to instances of any descendant concrete class and leverage polymorphism.

## Problem 13.3

3. Busy waiting is a method whereby a task waits for a given event by continuously checking for that event to occur. What is the main problem with this approach?

Inefficiency – the time spent by the task polling for the other event could be more constructively used by other tasks that are not waiting for anything.

## Problem 13.6

6. Suppose two tasks, A and B, must use the shared variable `Buf_Size`. Task A adds 2 to `Buf_Size`, and task B subtracts 1 from it. Assume that such arithmetic operations are done by the three-step process of fetching the current value, performing the arithmetic, and putting the new value back. In the absence of competition synchronization, what sequences of events are possible and what values result from these operations? Assume that the initial value of `Buf_Size` is 6.

There are 20 possible sequences (six events taken three at a time in which order counts). However, the only events that affect the results are when values are read and written, of which there are only six possible sequences (four events taken two at a time in which order counts).

	Sequence #1		Sequence #2		Sequence #3		Sequence #4		Sequence #5		Sequence #6	
Event	A	B	A	B	A	B	A	B	A	B	A	B
1	R(6)		R(6)		R(6)			R(6)		R(6)		R(6)
2	W(8)			R(6)		R(6)	R(6)		R(6)			W(5)
3		R(8)	W(8)			W(5)	W(8)			W(5)	R(5)	
4		W(7)		W(5)	W(8)			W(5)	W(8)		W(7)	
Buf_Size	7		5		8		5		8		7	

Thus there are three possible outcomes for the value of `Buf_Size`. This is not a general result, because some sequences yields the same results by coincidence. In general, Sequences #2 & #4 will always produce the same value; similarly, Sequences #3 & #5 will always produce the same value. But Sequences #1 and #6 will, in general, produce different values. They don't in this problem only because addition is commutative and associative.