Solutions to Exercises in Chapter 17

17.1 To be reusable, a class/package should be loosely coupled to other components, it should be portable, and it should not be too application-specific.

(a) A date class would be reusable, if implemented portably.

(b) A money class would be reusable.

(c) A matrix class would be moderately reusable: its use would be restricted to numerical applications.

(d) An employee class customized for a particular employer would not be reusable.

(e) A text package would be reusable.

(f) A word-bag class would be moderately reusable: its use would be restricted to text-processing applications.

(g) A generic bag class would be reusable.

(h) A family tree class would not be reusable: its use would be restricted to a narrow range of applications.

(i) A transportation network class would be moderately reusable: its use would be restricted to transportation applications.

(j) A text input-output package would reusable, if implemented portably (which is difficult).

(k) A graphics package would be reusable.

(l) A database management package would be reusable: it could be used in a wide variety of applications.

17.2 This exercise is about homogeneous and heterogeneous collections.

(a) A list of characters, each of type **`char`**, would be homogeneous.

(b) A stack of numbers, each of type **`int`** or **`float`**, would be heterogeneous.

(c) A stack of integers, each of type **`short`**, **`int`**, or **`long`**, would be heterogeneous.

(d) A list of objects, each of class `Date`, would be homogeneous (assuming that `Date` has no subclasses).

(e) A set of objects, each of class `Comparable`, would be heterogeneous: the `Comparable` interface is implemented by many classes. The smallest class that includes all possible elements is `Object`, but that class does not include *only* the possible elements.

(f) A set of objects, each of class `Student`, would be homogeneous (assuming that `Student` has no subclasses).

(g) A set of objects, each of class `Student` or `Staff`, would be heterogeneous. The smallest class that includes all possible elements is `Person`, but that class does not include *only* the possible elements.

(h) A set of objects, each of class `Student` or `Staff` or `Visitor`, would be heterogeneous. The smallest class that includes all possible elements is `Person`, and that class includes *only* the possible elements.

17.3 This exercise is about homogeneous and heterogeneous collection classes.

(a) A list of **`char`** values could be represented by either (i) one of the `List` classes in which the elements are `Character` objects, or (ii) a customized class in which the elements are char values. The only advantage of (i) is space. The advantage of (ii) is reuse. Usually (i) would be preferred.

(b) A set of **`char`** values could be represented by either (i) one of the `Set` classes in which the elements are `Character` objects, or (ii) a customized class similar to `IntSet` (Program 9.14). The advantages of (i) are time and space. The advantage of (ii) is reuse. In this case (ii) might be preferred.

(c) A list of arbitrary objects should certainly be represented by one of the `List` classes.

(d) A list of `Date` objects should certainly be represented by one of the `List` classes. The only (minor) advantage of a customized homogeneous list class would be that elements need not be cast to type `Date` when retrieved from lists.