Spring 2002 Dr. Ching

CALIFORNIA STATE UNIVERSITY, SACRAMENTO
College of Business Administration

MIS 211 - Information Systems II

H3 - Object-Oriented Programming and Arrays

Points: 25 Due Date: Wednesday, March 6

The objectives of this assignment are to familiarize you with arrays and object-oriented (OO) programming. It involves:

- Creating a class and subclass,
- Instantiating two objects in the subclass (binary search and sequential search),
- Loading data from the videos.dat file into three arrays (stock number, movie titles and prices),
- Sequentially accessing the arrays and printing their contents.

Specifications

Class and Subclass

Declare a class. The class members will include the following:

Array:	Define two arrays with 100 elements (each),
Variables:	Total number of elements used (integer) in each array and a subscript (integer).
Methods:	Initialize the counters (i.e., total number of elements used) to zero (constructor),
	Load Arrays - Assign values sequentially read from the videos.dat file to the arrays,
	Retrieve Arrays - Sequential retrieve values stored in each array,
Subclass:	Define a subclass of the class (the subclass will inherit all its data and methods from its class, hence it will contain nothing of its own),
Objects:	Instantiate two objects in the subclass (named): binary search, sequential search

Note. The movie titles will be stored in an array declared in the main function of the program. Due to difficulties in accessing character strings that are retained in an object, this approach although not OO poses the least programming problems (i.e., more parsimonious).

Input

The following describes the record layout of videos.dat (compressed in h2.exe):

Field	Data Type
Stock number	Double
Video title	String (40 characters)
Price	Double

Output

- Page and report heading. The page heading consists of a simple, descriptive title and column headings.
- Detail lines. Aesthetically arrange the columns over 80 columns (8½ x 11-inch page, portrait orientation). Each detail line will contain data retrieved from corresponding elements of the three arrays, such that the first detail line will contain data from the first elements of the stock number, movie title and price arrays, the second detail line will contain data from the second elements of the stock number, movie title and price arrays, etc.
- Report footing. The total number of records read from the file. This should be identical to the total number of elements used in the array.
- Aesthetic design. Aesthetically arrange the heading, detail lines and footing in your report. These include (but are not limited to) column alignment, report title and column heading placement.

Note. When printing your report, use a non-proportionate font.

Processing

There are two primary processes: sequentially loading the arrays with data read from videos.dat, and sequentially retrieving data from the arrays. Before these can be done, two objects belonging to the subclass must be instantiated.

Instantiating the Objects

Instantiate two objects in the (single) subclass: binary search and sequential search (searches will be conducted on these arrays in H4). The subclass will inherit its variables and arrays from its class.

Loading the Arrays

Sequentially read the videos.dat file, and load the data from each record into an element in their respective arrays (stock numbers, movie titles, prices). Do this until the end of file (EOF) is reached. Because movie title is composed of a character string, you will need to have three "reads." For example, if you have assigned the videos.dat file an alias (i.e., logical name) of "inputfile," the three reads will appear as such:

```
inputfile >> stock-number-variable-name(subscript);
inputfile.getline(movie-title-variable-name,41);
inputfile >> price-variable-name;
```

The *getline* overcomes the problem of spaces being used as delimiters. Without the *getline*, each word would be interpreted as a single value rather than a string.

Because the stock number and price arrays reside in the objects, you will need to use a method to move their values into the arrays (only one methods is needed for this). Be sure to increment the subscript each time you read a record and place its values into the arrays (i.e., the first record's data goes into the first elements, the second into the second element's, etc.).

Loading the movie titles into the movie title array requires accessing a new library (preprocess directive), and the use of STRCPY (string copy). Add *string.h* to the list of directives in your program's main function (e.g., `#include <string.h>`). To load the movie title into the array, use the following syntax:

`strcpy(`*target*,*source*`)`

Where:	*target*	The array name including the subscript (e.g., movie_title[i], where *i* is the subscript).
	source	The variable name in the *getline*.

Example: `strcpy(movie_titles[i],movie_title)`

Retrieving Data from the Arrays

Sequentially retrieve data from the elements of the three arrays and write them to a report file. Be sure to synchronize the subscripts so the elements corresponding to the same locations are accessed at the same time (i.e., first elements of the stock number, movie title and price arrays, second elements of the stock number, movie title and price arrays, etc.). One detail line will contain the data from the elements corresponding to the same locations (in the arrays). Because the stock number and price arrays reside in objects (binary search and sequential search), a method must be used to retrieve them.

Note. Retrieving the contents from one of the arrays will be sufficient. Be sure both arrays have been loaded, though.

Tangibles

Submit the following items in a 9 x 12-inch manila envelope:

- Hardcopies of both the output (e.g., report.txt) and program (i.e., source listing).

  Note. Please assign a txt extension to your output file. This makes it easier to open your file in Notepad.

- 3 ½-inch diskette with your entire project (workspace) folder, including the program (ccp), data file (dat) and output file (txt).