****

Author: Matthew Pfister
Project: C++ Painting Manager (extended)
Class: CS474
Date: 04/30/13

****

>How to compile

Included in the package is a makefile:
To compile the program type the command, make, in the bash.
To execute the program type the command, ./paintingmanager, in the bash.
To remove the binary file type the command, make clean, in the bash.

>How to use the program

Valid commands:
a (add artist) → The program will ask for the user to input the name of the artist, first and last. If the artist is not currently in the list of artists, or the list is empty, a new artist node will created with the given name. Otherwise, the program will output an error and return to the main menu.

p (add painting) → The program will ask for the user to input the name of the painting's artist as well as the title, width, and height. In addition, it will ask for they type of the painting being input (landscape, portrait, or stilllife). Depending upon which type is chosen, more information will need to be filled out (prompts from the program will tell the user what additional information to input). If the artist does not exist already, it will create a new list of paintings for that artist and append the new painting to it. If the artist does exist, it will check to see if the datastructure already contains a painting with the same name. If it does, it will output an error and then return to the main menu. Otherwise, it will append the new painting to the end of the artist's list.

d (delete artist) → The program will ask for the name of the artist to delete. If the artist does not exist it will output an error, and return to the main menu. Otherwise, the program will recursively delete all the paintings stored under that artist's name, and then remove the artist from the list.

r (remove painting) → The program will ask for the name of the painting's artist, as well as the id number (The program requests the artist's name so it doesn't have to traverse every artist's painting list). It will then jump to that artist and traverse the list of paintings associated with him/her. If it finds a matching painting with the same id, it will remove it. Otherwise, it will output an error and return to the main menu.

c (copy artist) → The program will ask for the name of the artist that the user wishes to copy. If the artist exists the program will then prompt the user for a new first name for the copy. Finally, the program preforms a deep copy and appends the newly created artist at the end of the artist list. If the artist is not found initially, the program will output an error and return to the main menu.

k (copy painting) → The program will ask for the name of the painting's artist, as well as, the id number (the program requests the artist's name so it doesn't have to traverse every artist's painting list). It will then jump to that artist and traverse the list of paintings associated with him/her. If it finds a matching painting with the same id, it will preform a deep copy and append a "_COPY(#id number)" tag to the end of the new copy's title. Otherwise, it will output an error and return to the main menu.

l (list all paintings) → The program will traverse through every artists' list of paintings and print out their titles (after printing the artist's name of course). More importantly, it will also print out id numbers and all varying information, such as: location for landscapes, medium for stilllifes and the number and names of people in a portrait. In addition, it will print out the total number of DIFFERENT artists that are stored in the datastructure.

q (quit) -> Exits the program.

>Misc

When entering a new painting, if anything is entered into the width and height fields, other than a number, the program will output an error and will prompt the user to try again.

When entering a command in the main menu, if anything is entered, other than one of the commands listed above, the program will output an error and will prompt the user to try again.

Some of the files listed are ArtistList, PaintingList, Artist, and Painting. Each is its own class, and the two lists are Linked Lists that hold their respected node types (i.e. ArtistList → Artist and PaintingList → Painting).

Landscape, Portrait and Stilllifes are all derived classes of painting and have their own copy constructors and destructors.