EGC-211
C++ Lab Assignment 5
Sept 18, 2024

Due date and time: Sept 25, midnight

Background: Here is your chance to attempt the in-class programming test again with certain additional features and requirements. It is recommended that you write this program afresh following the discussion on using virtual functions to solve this problem.

Additional Requirements: Since this is your only chance to try out the last few features of C++ taught in class, here is a list of additional requirements for this assignment:

1. Print() function must be implemented as a virtual function.
2. Use only ONE vector<> or list<> from STL to store (pointers to) all categories of species. No C-style arrays should be used. Do not assume any limit on the number of species to be added under the 4 categories.
3. Virtual destructor is required in the base class.
4. On exit, you must save the current database of species from memory to a file called "db.txt" in text format with the same structure as printed by the "show A" command. Previous contents (if any) of "db.txt" must be discarded.
5. On start, you must open a file called db.txt from local disk and load all the records saved by your program on its previous run into memory. Before the first run, you can create an empty file called "db.txt" because there will not be any data on the first run.
6. Any add / delete / show commands should build on the previous state of the data that you load from db.txt on start.
7. You must use ONE exception along with try-throw-catch in your program, e.g. if you do not find db.txt on starting the program.
8. You must submit a makefile along with your code that compiles your program.
9. Error messages must specify what kind of error was encountered and provide correct usage.

Assignment: Design a system to track zoo animals so appropriate food can be ordered for them. For simplicity, we will not track individual animals but rather track each species along with the count of individuals in that species, e.g. 6 langoors, 2 lions, 1 elephant, etc.

1. Implement the following C++ classes according to the description given:

   i) Zoo_species is an "abstract" super-class (see note 4 towards the end), containing the following members:
      a. Species_name – protected attribute
      b. Species_count – protected attribute
      c. Set_name(string name) – public method to set Species_name
      d. Set_count(int count) – public method to set Species_count
      e. Print() – public method to print name and count

ii) Following 4 category classes are derived from Zoo_species class:

a. Mammal class, which defines
   i. an additional private attribute called “Diet_type” of enum diet_type with 2 values {herbivore, carnivore},
   ii. a public method to set Diet_type called “Set_diet_type()” that takes an enum diet_type argument
   iii. Print() method for mammals should also print the Diet_type value along with name and count, e.g. “deer 20 herbivore”

b. Reptile class, which defines
   i. An additional private attribute called “Feed_size” of enum size_type with 3 values {small, medium, large}
   ii. A public method to set Feed_size called “Set_feed_size() that takes an enum size_type argument
   iii. Print() method for reptiles should also print the Feed_size value along with name and count, e.g. “crocodile 2 large” (indicating that crocodiles need a large size feed)

c. Bird class, which defines
   i. An additional private attribute called “Bird_feed” of enum bird_feed_type with 3 values {grain, insect, fish}
   ii. A public method to set Bird_feed called “Set_bird_feed() that takes an enum bird_feed_type argument
   iii. Print() method for birds should also print the Bird_feed value along with name and count, e.g. “kingfisher 4 fish”

d. Aquatic class, which defines
   i. An additional private attribute called “Aqua_feed” of enum aqua_feed_type with 3 values {fishfood, livefish, plants}
   ii. A public method to set Aqua_feed called “Set_aqua_feed() that takes an enum aqua_feed_type argument
   iii. Print() method for aquatics should also print the Aqua_feed value along with name and count, e.g. “hippo 2 plants”

2. Your program should indefinitely wait for commands until given an “exit” command to exit. Implement the following commands for zoo species:

   i) **add** command for adding a species along with its category (M – Mammal, R – Reptile, B – Bird, Q – Aquatic) and special values, with the following syntax:
   add <category-name-char> <species-name-string> <count> <category-specific-attribute-string>

   Every add command for a new species should result in a new object being created of the appropriate category class and initialized with species name, count and category specific attribute. Note that there are no double quotes around string inputs, which means that species names must be limited to single word strings, e.g. lion and not “white lion” or eagle and not “bald eagle”. Adding the same species name multiple times is allowed as long as the values of category-specific attribute match. In this case, you will simply add the new count to the existing count. Attempting to add count to an existing

species should fail with an error if the category-specific attribute is not the same. Negative or 0 count in add command should result in an error.

Examples:

a. add M lion 2 carnivore
   creates a new object of class Mammal with Species_name = lion, Species_count = 2 and Diet_type = carnivore
b. add M lion 2 herbivore
   after adding 2 lions previously as carnivore will result in an error
c. add M lion 2 carnivore
   second time simply adds the new count to existing count making it 4 lions without changing the count of mammals or total species.
d. add R iguana 4 medium
   creates a new object of class Reptile with Species_name = iguana, Species_count = 4 and Feed_size = medium
e. add B eagle 2 insect
   creates a new object of class Bird with Species_name = eagle, Species_count = 2 and Bird_feed = insect
f. add Q hippo 2 plants
   creates a new object of class Aquatic with Species_name = hippo, Species_count = 2 and Aqua_feed = plants.

ii) **delete** command is used to remove some number of species from the zoo with the following syntax:
delete <category-name-char> <species-name-string> <count>

Note that category-specific-attribute is not needed in the delete command. If the specified species name exists under the specified category and its existing count is greater than or equal to the count specified by delete, the existing count for that species is decremented by the number specified in delete. If the count for a species reaches 0, the species must be deleted after decrementing the category count and total species count. Trying to delete a species that was not added or trying to delete a number greater than the existing count for a species should result in an error with no change in data. Giving a negative or 0 count in the delete command should give an error.

Examples:

a. delete M lion 2
   decrements the lion count under mammals by 2. If the count reaches 0, lion species is deleted decrementing mammal and total species count by 1.
b. delete M tiger 3
   if there were no tigers added or if the count of tigers was less than 3, this will result in an error.

iii) **show** command to print all species added to a specific category or all categories (specified by character 'A') with the following syntax:
show <category-name-char>

Examples:

a. show M
   prints the number of mammal species added so far on line 1 (0 if none were added) followed by a list of added mammal species with their names, counts and special attributes, one species per line – in the order they were added, e.g.

   mammal 3
   lion 2 carnivore
   deer 34 herbivore
   langoor 12 herbivore

b. show R
   prints the number of reptile species added so far on line 1 (0 if none were added) followed by a list of added reptile species with their names, counts and special attributes, one species per line – in the order they were added, e.g.

   reptile 2
   crocodile 1 large
   iguana 1 medium

c. show B
   prints the number of bird species added so far on line 1 (0 if none were added) followed by a list of added bird species with their names, counts and special attributes, one species per line – in the order they were added, e.g.

   bird 1
   kingfisher 2 fish

d. show Q
   prints the number of aquatic species added so far on line 1 (0 if none were added) followed by a list of added aquatic species with their names, counts and special attributes, one species per line – in the order they were added, e.g.

   aquatic 1
   hippo 2 plants

e. show A
   prints
   i. the total number of species added so far on line 1 (0 if none were added), followed by
   ii. all mammal species with their names, counts and special attributes as shown above in example a, followed by

iii. all reptile species with their names, counts and special attributes as shown above in example b, followed by
iv. all bird species with their names, counts and special attributes as shown above in example c, followed by
v. all aquatic species with their names, counts and special attributes as shown above in example d, e.g.

```
total species 7
mammal 3
lion 2 carnivore
deer 34 herbivore
langoor 12 herbivore
reptile 2
crocodile 1 large
iguana 1 medium
bird 1
kingfisher 2 fish
aquatic 1
hippo 2 plants
```

f. show A
will always print the total number of added species followed by species in each category in the order of i) mammals followed by ii) reptiles, followed by iii) birds. Within each category, the order of addition will be followed for printing, e.g. if lion species was added before deer, it will be printed before deer as well. If no species were added, this command will print:

```
total species 0
mammal 0
reptile 0
bird 0
aquatic 0
```

iv) **exit** command saves the existing species into db.txt in the same format and structure as the output of "show A" and exits the program with 0 return value.

3. Error handling: wrongly formatted commands should result in an error. Since we are not submitting this assignment to DOMjudge, you need to provide error messages that convey the kind of error encountered and ways to fix it. Similarly, supplying wrong category-specific attribute value should result in an error, e.g. adding a bird with "herbivore" as an attribute. We will not check if nonsensical attributes are added for a species as long as they are within the enum defined for a category, e.g. adding lion as herbivore mammal or monkey as carnivore mammal is allowed as we are not doing any semantic checking here.

4. You may add additional attributes and / or methods to any class if you like but you must implement the attributes and methods that are listed for each class.

5. You may use vector, list or any other STL classes with iterators if you wish but all the species must be stored in a generic container that holds pointers of type (Zoo_species *)

6. Appropriate constructors are expected for all classes.

7. Destructors may be defined for derived classes if additional memory is allocated within each object for data members. Virtual destructor is required for the base class.

8. Submit your program to LMS only within specified time. Include a makefile that compiles your program and any input files that contain commands to be fed into the program for testing. Make sure your program compiles and behaves correctly on Sample inputs. Additional test cases will be run during evaluation.