CS-224 Object Oriented Programming and Design Methodologies

Homework 05

Fall 2022

1 Guidelines

You need to submit this homework on 15th November, on LMS. Late submissions are allowed until 17th Nov 11:59pm, which will be penalized by 20%. Your work will not be accepted once the submission is closed on LMS.

- You need to do this assignment in a group of two students.
- You will submit your assignment to LMS (only one member of the group will submit).
- Clearly mention the group composition in submitted file name e.g. AhmadHassan_ah01345_BatoolAiman_ba03451.zip.
- You need to follow the best programming practices
- Submit assignment on time; late submissions will not be accepted.
- Some assignments will require you to submit multiple files. Always Zip and send them.
- It is better to submit incomplete assignment than none at all.
- It is better to submit the work that you have done yourself than what you have plagiarized.
- It is strongly advised that you start working on the assignment the day you get it. Assignments WILL take time.
- DO NOT send your assignment to your instructor, if you do, your assignment will get ZERO for not following clear instructions.
- You can be called in for Viva for any assignment that you submit

2 HUMania++

This is an extension to HUMania developed in HW4. Now you have to improve it by implementing inheritance, and showing polymorphism.

A sample code is given in HUMania folder, if you run it you can see a pigeon is drawn. This example creates just one object of Pigeon to show how things are drawn in SDL. Refer to `Pigeon.hpp/cpp and HUMania.cpp` ⇒ `drawObjects()`.

You are required to:

- Create a class `Unit`, which deals with drawing of an object. It has a `draw` method, completely implemented, and it draws one of the three states of object. It also have `fly` method, since fly behavior is different for different types of objects, therefore it's not fully implemented rather it's made `virtual`.

- Create a `Pigeon` class inherited from `Unit`. It over-rides the `fly` function, that flies the pigeon gradually to right side, and rotates through the screen.

- Create a `Butterfly` class inherited from `Unit`. It over-rides the `fly` function, that should take the butterfly right-down direction. Once a butterfly reaches to bottom of the screen, it starts flying right-up direction. Once it reaches top of the screen it moves right-down. Similar to the pigeon, it should rotate through the screen.

- Create a `Bee` class inherited from `Unit`. It over-rides the fly function, that should make it fly towards right only. During fly it should hover (doesn't move forward) for a while over a random interval. You may choose 5% probability in every frame to decide whether it starts hover, and it keeps hovering for 10 frames. As a bee reaches to right most border of screen, it exits from the game, hence the object must be removed from the bees vector properly.

- Every object animates three of the images provided in the assets file. The draw function is only drawing the object.

- As you click on the screen, one of the above objects is created randomly. Let's create a class `ObjectCreator`, it has only one function `Unit* getObject()`. This function creates one of the objects (Pigeon, Butterfly, Bee) randomly, and returns the pointer to that object. You'll maintain just a single list of `Unit*` to store objects of all the types. Simply iterate over the list and call draw and fly functions of each object.

- You have to create objects dynamically with `new` operator, hence the list should hold pointers to all of the objects. Remember to delete the objects when game is ended, and when the bee objects are removed from vector.
- As you are maintaining a single list to store all types of objects, therefore it would be essential to find out the Bee objects, because it's required to remove Bee objects when they leave the window. For this task you have to identify the type of objects. Please refer to this page to find type.
- Please refer to `Solution.exe` file to see it all in action.
- Are you having fun?? You are more than welcome to add more stuff to make this game interesting, e.g. some natural random movement of butterflies, sitting them on ground, pigeons sweeping etc. [*It doesn't carry any marks*]

3 `std::list` Tutorial

list is an std library's doubly linked list implementation. It provides constant time insertion and deletion operations, but it doesn't support indexing. You can look for complete manual here: `https://en.cppreference.com/w/cpp/container/list`.

```
#include<iostream>
#include<list>

using namespace std;

class Distance{
   int feet, inches;
   public:
   Distance(int ft, int inch): feet(ft), inches(inch){}
   void show(){
      cout<<feet<<"'"<<inches<<"\""<<endl;
   }
};

int main(){
   list<Distance*> dst; // It's a list that can store Distance type
       objects
   dst.push_back(new Distance(3, 4)); // create an object, and push
       it in vector
   dst.push_back(new Distance(5, 2));
   dst.push_back(new Distance(2, 7));
   dst.push_back(new Distance(7, 8));
   dst.push_back(new Distance(13, 1));

   for(Distance* d: dst){ // list doesn't support indexing, but we
       can use python's style for loop.
      d->show();
   }

   // deleting the objects, need to delete every single object
       created dynamically
   for(Distance* d: dst){
      delete d;
   }

   dst.clear(); //clears all the items from vector
```

```
}
/////////////// Output: /////////////////
3'4"
5'2"
2'7"
7'8"
13'1"
```

---

4 Some important points:

- Sample code is there for your benefit. If you are going to use it, understand how it works.
- You do not need to follow the code given exactly. You can make changes where you see fit provided that it makes sense.
- Make the class declarations in hpp files, and provide function implementations in cpp files. Don't use hpp files for implementation purposes.
- A tutorial given here to remove the elements from vector, you might need it to remove bees as they exit the screen.
- As a general rule, class's deta is private, and functions are public. Don't use getter/setter functions to manipulate data, rather think in object oriented directions and provide all the functionality in the class.
- Complete reference for C++ list is given here `https://en.cppreference.com/w/cpp/container/list`
- You need to define separate `*.hpp` and `*.cpp` files for all the classes.
- Exact x,y,w,h values for images in assets file can be found by `http://www.spritecow.com/`.
- A tutorial for file I/O is given `http://www.cplusplus.com/doc/tutorial/files/`.
- You should take `www.cplusplus.com` and `www.cppreference.com` as primary web source to search about C++
- You have to follow best OOP practices as discussed in lectures.

5 How to compile

Open the given `HUMania` folder in vscode by choosing `File` ⇒ `Open Folder`. The game can be run by simply pressing F5 from vscode. If due to some reason it doesn't work, then go compiling and running it from terminal, as explained in `how to compile.txt`

6 Rubric

OOP Concepts	Inheritance and polymorphism is well implemented	5
Memory	Dynamic memory management is done properly	3
Functionality	All the functionality is implemented as described above	2
Total		10

Table 1: Grading Rubric