EEEC10008(515169) S23: Object-Oriented Programming

Polymorphism

What you will learn from Lab 9

In this laboratory, you will learn the concept of polymorphism in object-oriented programming.

TASK 9-1 VIRTUAL DESTRUCTOR

```
// lab9-1.cpp
#include <iostream>
using std::cout;    using std::endl;

class Point2D
{
private:
    int *x;
    int *y;
public:
    Point2D(){x = new int (0);y=new int (0); cout << "New X and Y" <<
endl;}
    ~Point2D(){delete x; delete y; cout << "Delete X and Y" << endl;}
};
class Point4D : public Point2D
{
private:
    int *z;
    int *t;
public:
    Point4D() : Point2D()
       {z = new int (0); t = new int (0);cout << "New Z and T " << endl;}
    ~Point4D(){delete z; delete t; cout << "Delete Z and T" << endl;}
};
int main()
{
    Point2D *pt = new Point4D;
    delete pt;
    return 0;
}
```

- The program above will produce the output:

  `New X and Y`

  `New Z and T`

  `Delete X and Y`

- Note that, it is valid to create a `Point2D` pointer by `Point4D` object since every `Point4D` is a `Point2D`. It is invalid to write as "`Point4D *pt = new Point2D;`"
- In this example, proper cleanup is not achieved here. That is because the `new` operator constructed an object of type `Point4D`, but the `delete` operator cleaned up an object of type `Point2D` pointed to by `pt`.

- Please declare the destructor of the base class `Point2D` to be `virtual`, and execute the program again.

TASK 9-2 VIRTUAL FUNCTION

✓ Virtual destructor provides run-time polymorphism to proper cleanups. Similarly, virtual functions are also used to allow run-time polymorphism as the following example.

```
// lab9-2.cpp
#include <iostream>
using std::cout;    using std::endl;

/*  The Point2D and Point4D defined in lab9-1                              */
/*  Add declarations of display() in Point2D and Point4D, respectively. */

void Point2D::display() const
{
    cout << *x << "," << *y;
}
void Point4D::display() const
{
    Point2D::display();
    cout << "," << *z << "," << *t;
}

int main()
{
    Point2D *pt = new Point4D;
    pt->display(); cout << endl;
    delete pt;

    return 0;
}
```

- The above program shows "`0,0`" for `pt` on screen. However, `pt` is created by object of type `Point4D`, and the expected result should be "`0,0,0,0`".
- Please declare the `display` of the base class `Point2D` as a virtual function, and execute the program again.
- In general, a class with a virtual function should have a virtual destructor, because run-time polymorphism is expected for such a class.

TASK 9-3 ABSTRACT CLASSES

✓ A virtual function is called a pure virtual function if it is declared but its definition is not provided. A class with one or more pure virtual functions is called an abstract class.

```
// lab9-3.cpp
#include <iostream>
using std::cout;   using std::endl;
```

```
class Shape
{
protected:
    int color;
public:
    virtual void draw() = 0;
    virtual bool is_closed() = 0;
    virtual ~Shape(){}
};
int main()
{
    Shape s;
    return 0;
}
```

- The object of an abstract class cannot be defined. It is illegal to define as "`Shape s;`"
- Some virtual functions in a base class can only be declared but cannot be defined, since the base class may not have enough information to do so and such virtual functions are only meant to provide a common interface for the derived classes.

✓ Class `Circle` is derived from the abstract class `Shape`.

```
// lab9-3-1.cpp
#include <iostream>
using std::cout;   using std::endl;

/* abstract class Shape defined in lab9-3  */
/* general class Point2D defined in lab9-2  */

class Circle: public Shape
{
private:
    Point2D center;
    double radius;
public:
    // constructor of Circle.
    void draw();
    bool is_closed() {return true;}
};
int main()
{
    Point2D pt(3,4);
    Circle c(pt,5,255);
    c.draw();

    return 0;
}
```

✓ Class `Polygon` is also an abstract class since it has pure virtual functions `draw()`, which is inherited from its base class `Shape` but has not been defined. Thus, objects of `Polygon` cannot be created. However, `Triangle` is no longer an abstract class since it does not contain any pure

virtual functions.

```
// lab9-3-2.cpp
#include <iostream>
using std::cout;   using std::endl;

/* abstract class Shape defined in lab9-3  */
/* general class Point2D defined in lab9-2  */

class Polygon: public Shape
{
public:
    bool is_closed() {return true;}
};
class Triangle: public Polygon
{
private:
    Point2D *vertices;
public:
    // constructor for Triangle
    ~Triangle() {delete [] vertices;}
    void draw();
};

int main()
{
    Point2D *vec = new Point2D[3];
    vec[0].setPoint2D(1,1);
    vec[1].setPoint2D(6,1);
    vec[2].setPoint2D(1,8);

    Triangle t(vec,255);
    delete []vec;

    t.draw();


    return 0;
}
```

EXERCISE 9-1: ELECTRONICS DEVICE

- In this problem, you should implement three classes. The base class is `Electronics` and the other classes, `Phone` and `TV`, are the derived classes. Following class template is for your reference. You should determine whether the function is virtual function or not.
- There are some files in `/home/share/lab9` folder for your information.
- `Electronics.h`

```
#ifndef ELECTRONICS_H_
#define ELECTRONICS_H_

using namespace std;

class Electronics{
   private:
   int length;
   int width;
   bool usage;
   bool charging;

public:
   Electronics(int length, int width);
   bool get_usage();
   void set_usage(bool flag);
   bool get_charging();
   void set_charging(bool flag);
   void charge();
   void poweron();
   void poweroff();
   void run();
};

#endif //ELECTRONICS_H_
```

- Class `TV`
  - No other data member
- Class `Phone`
  - Data member
    - int current_power
    - int in_rate
    - int out_rate
  - Constructor
    - Phone(int in_rate, int out_rate, int length, int width)
- `TV` can only be used when it is charging.
- When `Phone` call function `run()` and it is charging, `current_power` increases by `in_rate`.
- When `Phone` call function `run()` and it is usage, `current_power` decreases by `out_rate`.
- `current_power` cannot be lower than 0 and higher than 100.

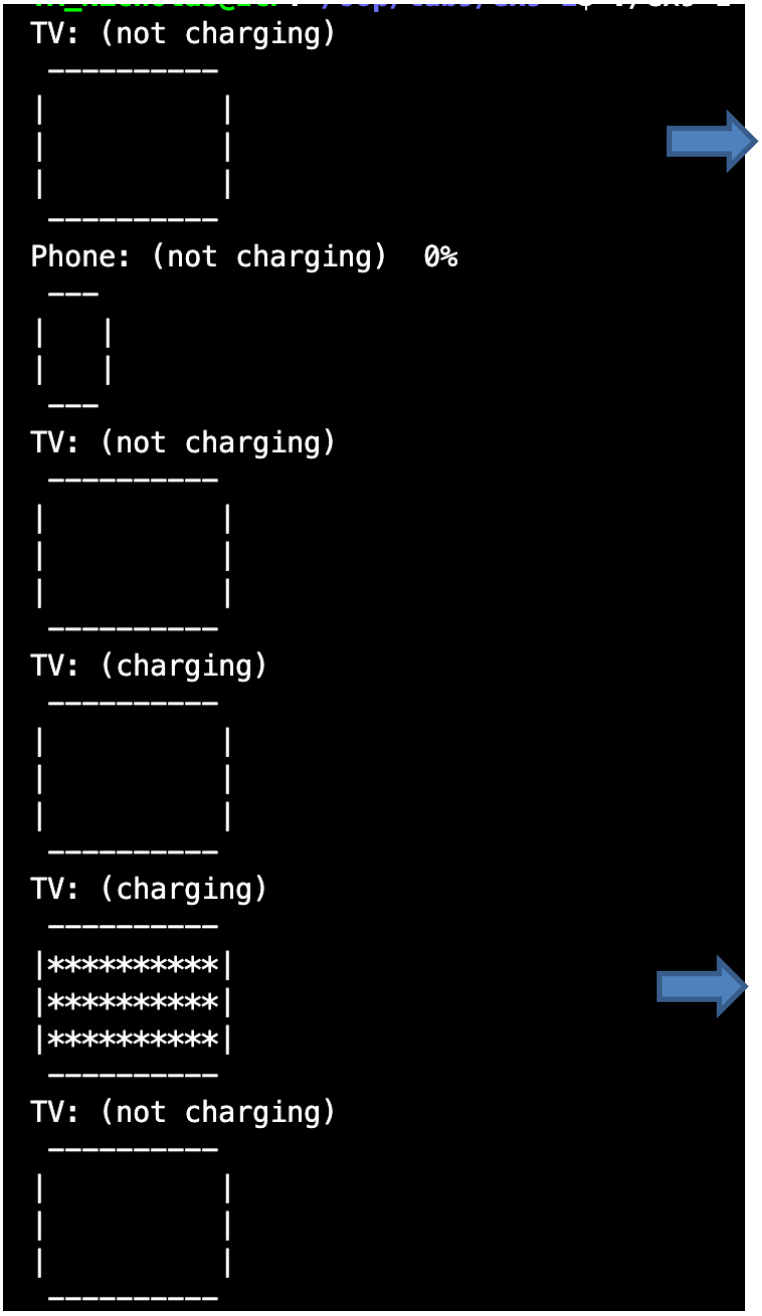
✓ Execute command: `>./ex9-1`

✓ Here is the main function and output sample.

`//ex9-1.cpp`

```
int main()
{
    Electronics* E1;
    Electronics* E2;
    E1 = new TV(5, 10);
    E2 = new Phone(50, 40, 4, 3);
……
```

`// output sample`



When usage = false

When usage = true

```
Phone: (not charging)  0%
 ---
|   |
|   |
 ---
Phone: (charging)  50%
 ---
|   |
|   |
 ---
Phone: (charging)  100%
 ---
|   |
|   |
 ---
Phone: (charging)  100%
 ---
|***|
|***|
 ---
Phone: (not charging)  60%
 ---
|***|
|***|
 ---
Phone: (not charging)  20%
 ---
|***|
|***|
 ---
Phone: (not charging)  0%
 ---
|   |
|   |
 ---
Phone: (charging)  10%
 ---
|***|
|***|
 ---
Phone: (charging)  60%
 ---
|   |
|   |
 ---
```

EXERCISE 9-2: EVALUATE COMMODITY

- In this problem, you should implement three classes and the main function. The base class is `Commodity`. And the other classes, `Food` and `Healthy`, are the derived classes.
- `Commodity.h`

```
#ifndef COMMODITY_H_
#define COMMODITY_H_

#include <string>
using namespace std;

class Commodity
{
private:

public:
  string name;
  double* score;
  int* price;
  Commodity(string name, int price);
  virtual ~Commodity();
  virtual void cal_score() = 0;
  virtual void show_spec() = 0;
};

#endif //COMMODITY_H_
```

- `Food.h`

```
#ifndef FOOD_H_
#define FOOD_H_

#include "Commodity.h"
using namespace std;

class Food : public Commodity{
private:
  int* car_value;
  int* pro_value;
  int* fat_value;
public:
  (constructor/destructor/other function……)
};

#endif //FOOD_H_
```

- `Healthy.h`

```
#ifndef HEALTHY_H_
#define HEALTHY_H_

#include "Commodity.h"
using namespace std;
```

```
class Healthy : public Commodity{
private:
  string* name_arr;
  int* value_arr;;
  int arr_length;
public:
  (constructor/destructor/other function……)
};

#endif //HEALTHY_H_
```

- ✓ Score:
  - Food: pro_value / price
  - Healthy: summation of all ingredient value / price
- ✓ In the main function, you should read the information from the input file and store it into a vector that contains the `Commodity pointer`. After the input is read, the vector should be sorted by the commodity's name. At last, call the function `show_spec()` in the order of vector. And don't forget to free the memory you use.
- ✓ If you don't know the fast way for sorting a vector, the example 4 in this website is for your information.

  https://shengyu7697.github.io/std-sort/
- ✓ In the input file, one line's information represents one commodity. In each line:
  - The first one is a char and it is 'H' or 'F'.
  - The second one is the commodity's name.
  - The third one is the commodity's price.
  - For a `Healthy`
    - The fourth one means the ingredient's number.
    - The rest of line are many pairs of ingredient's name and value.
  - For a `Food`
    - The rest of line are values of car, pro, fat.
- ✓ Execute command: `>./ex9-2 1.txt`
- ✓ To pass the test, your program cannot contain memory leaks. You can use `valgrind` to test for memory leaks.
- ✓ Following is `1.txt` and output sample

  `//1.txt`

```
H H1 20 2 A 10 B 35
H H2 10 5 A 10 B 10 C 10 D 10 E 10
F F1 5 1 2 2
F F2 10 3 5 2
```

```
//output sample
```

```
==1485347== Memcheck, a memory error detector
==1485347== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et
al.
==1485347== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright
info
==1485347== Command: ./ex9-2 1.txt
==1485347==
======================================
name: F1
price: 5
car: 1
pro: 2
fat: 2
score: 0.4
======================================
name: F2
price: 10
car: 3
pro: 5
fat: 2
score: 0.5
======================================
name: H1
price: 20
A: 10
B: 35
score: 2.25
======================================
name: H2
price: 10
A: 10
B: 10
C: 10
D: 10
E: 10
score: 5
======================================
==1485347==
==1485347== HEAP SUMMARY:
==1485347== in use at exit: 0 bytes in 0 blocks
==1485347== total heap usage: 37 allocs, 37 frees, 83,539 bytes allocated
==1485347==
==1485347== All heap blocks were freed -- no leaks are possible
==1485347==
==1485347== For lists of detected and suppressed errors, rerun with: -s
==1485347== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from
0)
```

✓ Please write every class in separate files and write a “Makefile” in your exercise directory to compile your code. Your code can be compiled by typing “`make`”, and the name of your

program should be `ex9-2`.