Классыг Загварчлан Удамшуулах, Эх Класс Болон Удамшсан Классуудын Харилцаа (Лаборатори №7)

Н. Энхболд

Програм хангамжийн 2-р түвшний оюутан, 20B1NUM0102

1. ОРШИЛ/УДИРТГАЛ

Объект хандалтат програмчлалын удамшил гэх ойлголтыг TwoDimensionalShape класс дээр шаталсан хэлбэрээр үзүүлсэн ба үүнээс Circle, Square, Triangle зэрэг хүүхэд классуудыг бий болгов. Удамшлын горимуудын түлхүүр үгүүдийг ашиглан Эх классын гишүүд рүү хандах Хүүхэд классын харилцааг харуулсан бөгөөд үүн дээр нэмэн удамшин шинээр үүссэн классын байгуулагч функц хэрхэн ажиллаж буйг хавсаргав.

2. ЗОРИЛГО

Shape классыг удамшуулан TwoDimensionalShape гэх классыг тодорхойлох бөгөөд энэхүү классаа дахин удамшуулан Circle, Square, Triangle зэрэг классуудыг тодорхойлон судалгааг хийх:

1. Удамшил
2. Удамшлын төрлүүд
3. Удамшлын давуу тал зэргийг тодорхойлон судлах зорилго тавив.

3. ОНОЛЫН СУДАЛГАА

3.1 Удамшил

Програмчлалын хэлүүдэд удамшуулах гэдэг ойлголт нь хэрэглэсэн зүйлээ дахин өөр зүйлд зориулан өргөтгөж хэрэглэх үед бий болдог. Бид ямар нэгэн зүйлийн ерөнхий шинжээр класс үүсгэн тухайн классаа илүү олон өгөгдөл бүхий тодорхой 1 зүйлд зориулан хэрэглэхдээ тухайн классаа удамшуулж шинэ класс үүсгэдэг. Удамшсан классыг “Эх класс” ба шинээр үүсч буй классыг “Хүүхэд класс” гэж нэрлэдэг.

3.2 Удамшлийн горимууд

Объектыг удамшуулахын тулд өмнөх эх классын гишүүн өгөгдлүүдийг/функцуудын “Хүүхэд классаас” хандах горимыг тодорхойлж өгдөг бөгөөд үүнийг удамшлын горим гэдэг. Эдгээр нь: public, private, protected зэрэг түлхүүр үгсээр зохицуулагдаж өгдөг.

- Public: Классыг энэхүү удамшлын горимоор удамшуулж өгснөөр “Хүүхэд класс”-аас эх класст буй гишүүдэд ямар нэгэн хандалтын горимын нэмэлт өөрчлөлтгүй шууд

байгаа горимоор нь хандаж болдог.

- Private гишүүдэд мэдээж шууд хандаж болохгүй
- Public гишүүдэд шууд хандаж болно
- Protected гишүүдэд зөвхөн “Хүүхэд класс”-аас шууд хандах боломжтой

- Private: Энэхүү горимоор удамшуулсан “Хүүхэд класс” “Эх класс” руугаа шууд хандах боломжгүй байдаг бөгөөд бүх гишүүд нь хандах боломжгүй private хэлбэртэй болж удамшдаг.
- Protected: Энэхүү горим нь харьцангуй уян хатан буюу гаднаас хандах шууд боломжгүй хэдий ч хүүхэд классаас хандах боломжийг бий болгодог
  - Public болон Protected гишүүдэд шууд хандах боломжтой
  - Private гишүүн рүү ямар ч горимоор удамшуулсан шууд хандах боломжгүй

3.3 Удамшлийн давуу болон сул тал

Давуу талууд:

1. Удамшлыг ашигласнаар ерөнхий ойлголт бүхий классыг элементарчлан шинэ класс үүсгэн өмнөх классын мэдээллийг өөрчлөлгүйгээр дахин ашиглах боломжийг бий болгодог.
2. Тухайн эх классаас үүдэлтэй олон хүүхэд класс үүсгэх бол удамшлыг ашиглан илүү бага кодын тусламжтайгаар бичих мөн ямар нэгэн эх классаас хамаарсан алдаа харьцангуй багасна.
3. Илүү бага зардлаар буюу энэхүү классаа ашиглан илүү бүтээмжийг ихэсгэх боломжтой.

Сул:

1. Хязгаарлагдмал буюу бид эх класст буй өгөгдлийг ашиглах шаардлага гараагүй үед энэ нь илүүц зардлыг бий болгоно.
2. Хэрэв бид олон хүүхэд класс үүсгэх шаардлагагүй тохиолдолд энэхүү эх классыг бий болгох нь ямар ч үр ашиггүй юм.

3.4 Удамшлын төрлүүд

Удамшлын 5 төрөл байдаг.

1. Нэг-нэг удамшил (Энгийн)
   - Ямар нэгэн хүүхэд класс зөвхөн 1 эх классаас удамших хэлбэр

```
Class Hvn{….};
Class Oyutan: public Hvn{…}
```

2. Нэг-олон удамшил (Нийлмэл)
   - Хүүхэд класс нь олон эх классаас удамшиж буй хэлбэр

```
Class Aav{…};
Class Eej{…};
Class Hvv : public Aaw, public Eej{…}
```

3. Олон түвшинт удамшил
   - Ямар нэгэн хүүхэд класс удамшин шинээр хүүхэд класс үүсгэх буюу давхар удамших хэлбэр

```
Class Mashin{…};
Class UraldaaniiMashin : public Mashin{..};
Class Bugatti: public UraldaaniiMashin{…};
```

4. Шаталсан удамшил
   - Нэг эх классаас олон хүүхэд класс үүсэх хэлбэр

```
Class Shape{…};
Class 2DShape : public Shape{…};
Class 3DShape : public Shape{…};
```

5. Холимог удамшил
   - Энгийн болон шаталсан зэрэг удамшуулах хэлбэрийн нийлбэр хэлбэр

```
Class Shape{..};
Class 2DShape : public Shape{..};
Class 3DShape : public Shape{…};
Class Ogtorgui : public 2DShape, public 3DShape{…};
```

4. ХЭРЭГЖҮҮЛЭЛТ

1. TwoDimensionalShape Классаас шаталсан хэлбэрээр Circle, Square, Triangle зэрэг хүүхэд классуудыг үүсгэв. Эх класст эдгээр хүүхэд класс бүрт байх name өгөгдлийг тодорхойлсон бөгөөд үүнийг динамик санах ойн new оператор ашиглан тохирсон хэмжээ бүхий санах ой нөөцлөх байдлаар загварчлав.

2. Constructor функцийг Эх класс болон Хүүхэд класст хоёуланд тодорхойлж өгсөн бөгөөд Хүүхэд классын Constructor нь Эх классын параметрт байгуулагчийг ашиглан гишүүн өгөгдөлдөө утга оноодог байдлаар загварчлав.

5. ДҮГНЭЛТ

Хүүхэд буюу үүсч буй класст байгуулагч тодорхойлон main() функц дотроо объект бий болгон ажлуулж үзэхэд тухайн хүүхэд классын эх класст буй байгуулагч давхар ажиллах бөгөөд энэхүү байгуулагч нь өөрийн гишүүн өгөгдлүүдээ үүсгэж байна. Мөн энэхүү 2 эх класс болон хүүхэд классын байгуулагчдын constructor-ууд дуудагдахад эх классын байгуулагч түрүүлж ажилласан бөгөөд өөрийн гишүүн өгөгдлүүддээ утга оноосны дараачаар хүүхэд классын байгуулагч ажиллаж эхлэв.

6. АШИГЛАСАН МАТЕРИАЛ

1. Объект хандлагат технологийн С++ програмчлал, Ж.Пүрэв, 2008, Улаанбаатар.

2. Объект хандалтат програмчлал – Лекц 08 - Удамшил

7. ХАВСРАЛТ

```
#include <iostream>

#include <math.h>

#define PI 3.14159


using namespace std;


class TwoDimensionalShape{

      protected:

            char *name;

            ~TwoDimensionalShape(){

                  delete name;

            }

      public:

            void setName(char *ner){

                  name = new char[strlen(ner) + 1];

                  strcpy(name, ner);

            }
```

```
            TwoDimensionalShape(char *ner){
                  name = new char[strlen(ner)+1];
                  strcpy(name, ner);
                  cout << "Dvrs vvssen" << endl;
            }
            TwoDimensionalShape(){
                  cout << "Dvrs vvssen" << endl;
            }
};

//class ThreeDimensionalShape{
//
//};

class Circle : public TwoDimensionalShape{
      private:
            float o[2];
            float r;
      public:
            float findArea(){
                  return PI * r * r;
            }
            float findCircle(){
                  return 2 * PI * r;
            }
            Circle() : TwoDimensionalShape("Dugui"){
            }
            Circle(char *ner, float x, float y) :
```

```
TwoDimensionalShape(ner){
                o[0] = x;
                o[1] = y;
          }
          void setRadius(float a){
                r = a;
          }
};

class Square : public TwoDimensionalShape{
     private:
          float a[2], b[2], c[2], d[2];
          float ab;
     public:
          float findArea(){
                return ab * ab;
          }
          float findFrame(){
                return 4 * ab;
          }
          Square() : TwoDimensionalShape("Kvadrat"){
          }
          Square(char *ner, float x, float y, float x1, float y1, float
x2, float y2, float x3, float y3) : TwoDimensionalShape(ner){
                a[0] = x;
                a[1] = y;
                b[0] = x1;
                b[1] = y1;
```

```
                c[0] = x2;
                c[1] = y2;
                d[0] = x3;
                d[1] = y3;
        }
        void setL(float l){
                ab = l;
        }
        void setA(float x, float y){
                a[0] = x;
                a[1] = y;
        }
        void setB(float x, float y){
                b[0] = x;
                b[1] = y;
        }
        void setC(float x, float y){
                c[0] = x;
                c[1] = y;
        }
        void setD(float x, float y){
                d[0] = x;
                d[1] = y;
        }
};
class Triangle : public TwoDimensionalShape{
    private:
        float a[2], b[2], c[3];
```

```
        float ab;
    public:
        float findArea(){
            return (ab * ab * sqrt(3) /2 /2);
        }
        float findFrame(){
            return 3 * ab;
        }
        Triangle() : TwoDimensionalShape("Gurvaljin"){
        }
        Triangle(char *ner, float x1, float y1, float x2, float y2,
float x3, float y3) : TwoDimensionalShape(ner){
            a[0] = x1;
            a[1] = y1;
            b[0] = x2;
            b[1] = y2;
            c[0] = x3;
            c[1] = y3;
        }
        void setL(float l){
            ab = l;
        }
        void setA(float x, float y){
            a[0] = x;
            a[1] = y;
        }
        void setB(float x, float y){
            b[0] = x;
```

```
                b[1] = y;
        }
        void setC(float x, float y){
                c[0] = x;
                c[1] = y;
        }
};

//class Sphere : public ThreeDimensionalShape{
//
//};
//
//class Cube : public ThreeDimensionalShape{
//
//};
//class Tetrahedron : public ThreeDimensionalShape{
//
//};

int main(){
      //Toirog
      Circle toirog("Dugui", 0, 0);
      toirog.setRadius(4);
      //Kvadrat
      Square kvadrat;
      int a[2] ={1, 1};
      int ab = 4;
      kvadrat.setA(a[0], a[1]);
```

```
kvadrat.setL(ab);
//B oroi
int b[2];
b[0] = a[0] + ab;
b[1] = a[1];
kvadrat.setB(b[0], b[1]);
//C oroi
int c[2];
c[0] = a[0] + ab;
c[1] = a[1] + ab;
kvadrat.setC(c[0], c[1]);
//D oroi
int d[2];
d[0] = a[0];
d[1] = a[1] + ab;
kvadrat.setC(d[0], d[1]);
//
Triangle gurvalj;
a[0] = 1;
a[1] = 0;
int l = 1;
gurvalj.setA(a[0], a[1]);
gurvalj.setL(l);
int y;
int x1, x2;
y = l * cos(30 * PI /180.00) + a[1];
x1 = l * sin(30 * PI/180.00) + a[0];
x2 = a[0] - l * sin(30 * PI/180.00);
```

```
        gurvalj.setB(y, x1);
        gurvalj.setC(y, x2);
}
```