TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM

KHOA CÔNG NGHỆ THÔNG TIN

BTC ÔN THI HỌC KỲ KHÓA 2016

Bài giải tham khảo

Đề thi cuối kì LTHĐT các năm 2014, 2015, 2016

- Người giải: Vương Hy – Lớp 16CNTN

cuu duong than cong . com

Cập nhật: 25/12/2017

Đề 2016

Câu 1:

```
Ellipse ellipse(2.0/3, 6);
```

⇨ Gọi contructor của lớp Ellipse nhưng => constructor của Ellipse gọi constructor của lớp cha là Shape

```
construct Shape
construct Ellipse
```

```
cout << "ellipse: " << endl;
```

```
ellipse:
```

```
cout << ellipse.Description() << endl;
```

In số thực thì sẽ in tối đa 6 chữ số có nghĩa (kể cả trước và sau dấu thập phân)

Do đã gọi phương thức shape->Scale(1.0/2); nên kích thước thay đổi

```
Ellipse(a=0.333333, b= 3)
```

```
cout << ellipse.Area() << endl;
```

```
3.14159
```



```
cout << ellipse.InterfaceType().name() << endl;
```

Do biến ellipse có kiểu Ellipse nên sẽ gọi hàm trong lớp Ellipse và in ra

```
Ellipse
```

```
cout << ellipse.ImplementationType().name() << endl;
```

Dù hàm được gọi là hàm của lớp Shape nhưng con trỏ this lúc này sẽ chứa đối tượng Ellipse nên sẽ in ra là Ellipse (tùy trình biên dịch như trong vs in ra là class Ellipse)

```
Ellipse
```

```
cout << "shape = &ellipse: " << endl;
```

```
shape = &ellipse:
```



```
cout << shape->Description() << endl;
```

Do hàm Description khai báo là hàm ảo (virtual) nên hàm Description của lớp Ellipse sẽ được gọi

```
Ellipse(a=0.333333, b= 3)
```

```
cout << shape->Area() << endl;
```

```
3.14159
```

cout << shape->InterfaceType().name() << endl; Do lớp phương thức InterfaceType không phải là hàm ảo nên phương thức của lớp Shape sẽ được gọi

Shape

cout << shape->ImplementationType().name() << endl; typeid sẽ trả về đúng kiểu đối tượng mà con trỏ this này đang trỏ tới

Ellipse

Hết hàm main => gọi destructor của các biến cục bộ theo thứ tự ngược lại với thứ tự khai báo. Ở đây chỉ có biến ellipse Destructor sẽ được gọi theo thứ tự từ lớp con đến lớp cha

destruct Ellipse

destruct Shape

Tóm lại output là:

construct Shape

construct Ellipse

ellipse:

Ellipse(a=0.333333, b=3)

3.14159

Ellipse

Ellipse

shape = &ellipse:

Ellipse(a=0.333333, b=3)

3.14159

Shape

Ellipse

destruct Ellipse

destruct Shape

- Shape::Description() và Circle::Description()
  - Đây là ghi đè (overriding) hàm, Circle::Description() override lại hàm Shape::Description() của lớp cha vì Description ở lớp Circle khác với lớp Shape nên ta không thể dùng lại phương thức của lớp cha hơn nữa đây lại là hàm ảo nên việc override này sẽ cho phép đối tượng Circle được quản lý bằng biến kiểu Shape * nhưng vẫn dùng phương thức Description của lớp Circle
- Ellipse::Scale(float) và Ellipse::Scale(float, float)

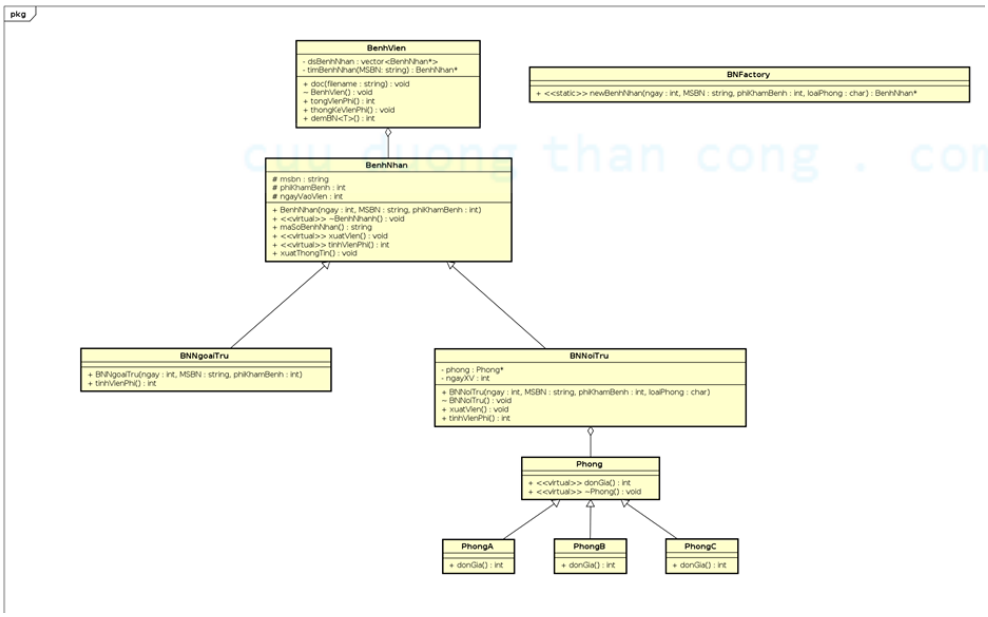
- Đây là nạp chồng, quá tải (overload) hàm, cùng một tên hàm nhưng lại có tham số truyền vào khác nhau thường overload hàm sẽ giúp ta có nhiều cách để thực hiện cùng một công việc cho nhiều kiểu tham số khác nhau. Như bài này Scale(float) chỉ có 1 tham số vào và sẽ thay đổi theo cả phương cùng một tỉ lệ, còn Phương thức Scale(float, float) sẽ có thể thay đổi trục chính và trục phụ theo 2 tỉ lệ riêng biệt.

- Shape::InterfaceType() và Circle::InterfaceType()
  - Đây cũng là ghi đè hàm (overriding). Nhưng lần này 2 phương thức này không phải là hàm ảo nên biến có kiểu nào sẽ gọi đúng phương thức ở lớp đó được xác định lúc dịch chương trình.
- Circle::InterfaceType() và Ellipse::InterfaceType()
  - Không có quan hệ gì

c) Dòng 88: Lỗi lớp Shape không có phương thức Scale nhận vào 2 tham số float
Dòng 102 + 103: Lớp Shape có chứa hàm thuần ảo (Area, Scale) nên là lớp thuần ảo do đó không thể tạo đối tượng có kiểu Shape
d) Dòng 88: ép kiểu động bằng cách (dynamic_cast<Ellipse *>shape)->Shape(3, 1.0/3)
Dòng 102, 103, có thể gọi tường minh trực tiếp shape->Shape::Description();

Câu 3:

Sơ đồ lớp:



Cài đặt

```
class BenhVien {
private:
	vector<BenhNhan*> dsBenhNhan;
	BenhNhan* timBenhNhan(string MSBN) {
		for (auto& x: dsBenhNhan)
			if (x->maSoBenhNhan() == MSBN)
				return x;
		return NULL;
	}
public:
	void doc(string fileName) {
		//cout << "asdfjasfajsgdfasdf";
```

```
		freopen(fileName.c_str(), "r", stdin);
		stringstream ss;
		string s;
		int ngay;
		while (cin >> ngay) {
			// cout << s << endl;
			int phiKhamBenh;
			string MSBN;
			string type;
			char loaiPhong = 0;
			cin >> MSBN >> type;
			if (type == "XV") {
				timBenhNhan(MSBN)->xuatVien(ngay);
				continue;
			} else if (type == "TKVP") {
				for (auto& x: dsBenhNhan)
					x->xuatVien(ngay);
				continue;
			}

			cin >> phiKhamBenh;

			if (type == "NV")
				cin >> loaiPhong;
			dsBenhNhan.push_back(BNFactory::newBenhNhan(ngay, MSBN, phiKhamBenh,
loaiPhong));
		}
	}
	~BenhVien() {
		for (BenhNhan*& x: dsBenhNhan)
			delete x;
	}
	int tongVienPhi() {
		int res = 0;
		for (BenhNhan*& x: dsBenhNhan)
			res += x->tinhVienPhi();
		return res;
	}
	void thongKeVienPhi() {
		cout << "Thong tin benh nhan:\n";
		for (BenhNhan*& x: dsBenhNhan) {
			x->xuatThongTin();
			cout << endl;
		}
		cout << "> Tong ket vien phi: " << tongVienPhi();
	}
	template<class T>
	int demBN() {
		int res = 0;
		for (auto& x: dsBenhNhan)
			res += !!dynamic_cast<T*>(x);
		return res;
	}
};

class BenhNhan {
protected:
	string MSBN;
```

```
	int phiKhamBenh;
	int ngayVaoVien;
public:
	BenhNhan(int ngay, string _MSBN, int _phiKhamBenh): ngayVaoVien(ngay),
MSBN(_MSBN), phiKhamBenh(_phiKhamBenh) {}
	virtual ~BenhNhan() {}
	string maSoBenhNhan() {
		return MSBN;
	}
	virtual void xuatVien(int) {}
	virtual int tinhVienPhi() {return 0;}
	void xuatThongTin() {
		cout << "Ma so benh nhan: " << MSBN << endl;
		cout << "Vien phi: " << tinhVienPhi();
	}
};

class BNNoiTru: public BenhNhan {
	Phong* phong;
	int ngayXV = -1;
public:
	BNNoiTru(int ngay, string _MSBN, int _phiKhamBenh, char loaiPhong): BenhNhan(ngay,
_MSBN, _phiKhamBenh) {
		ngayVaoVien = ngay;
		if (loaiPhong == 'A')
			phong = new PhongA;
		else if (loaiPhong == 'B')
			phong = new PhongB;
		else
			phong = new PhongC;
	}
	~BNNoiTru() {
		delete phong;
	}
	void xuatVien(int ngay) {
		if (ngayXV == -1)
			ngayXV = ngay;
	}
	int tinhVienPhi() {
		return (ngayXV - ngayVaoVien) * (phong->donGia() + phiKhamBenh) ;
	}
};

class BNNgoaiTru: public BenhNhan {
public:
	BNNgoaiTru(int ngay, string _MSBN, int _phiKhamBenh): BenhNhan(ngay, _MSBN,
_phiKhamBenh) {}
	~BNNgoaiTru() {}
	int tinhVienPhi() {
		return phiKhamBenh;
	}
};
class BNFactory {
public:
	static BenhNhan* newBenhNhan(int ngay, string MSBN, int phiKhamBenh, char
loaiPhong = 0) {
		if (loaiPhong)
			return new BNNoiTru(ngay, MSBN, phiKhamBenh, loaiPhong);
```

```
                else
                        return new BNNgoaiTru(ngay, MSBN, phiKhamBenh);
        }
};

class Phong {
public:
        virtual int donGia() = 0;
        virtual ~Phong() {}
};
class PhongC: public Phong {
public:
        int donGia() {
                return 600000;
        }
};
class PhongB: public Phong {
public:
        int donGia() {
                return 900000;
        }
};
class PhongA: public Phong {
public:
        int donGia() {
                return 1400000;
        }
};
```

Đề 2015

Câu 1:

a.

Kết quả in ra là

3 4

Giải thích:

Ở a.f(3) dù f là hàm ảo nhưng do A không phải là kiểu con trỏ nên chương trình vẫn sẽ gọi hàm f của A

Ở b.f(3) thì chương trình sẽ gọi hàm B::f (3). Hàm B::f(3) đó gọi hàm A::f(3 + 1) in ra 4

b.

Lớp A có cấp phát bộ nhớ và sử dụng biến con trỏ int * a nên phải thực hiện cài đặt 3 ông lớn là copy constructor, operator =, và destructor. Trong đoạn mã đề bài cho chỉ mới cài đặt destructor nên ta sẽ cài đặt thêm

```
class A {
public:
	A() { a = new int[3]; for (int i = 0; i < 3; i++) a[i] = i + 1; }
	~A() { delete[] a; }

	A(const A & obj) {
		a = new int[3];
		for (int i = 0; i < 3; i++) a[i] = obj.a[i];
	}

	A & operator =(const A & obj) {
		for (int i = 0; i < 3; i++) a[i] = obj.a[i];
		return (*this);
	}
private:
	int *a;
};
```

c)

```
class Singleton {
private:
	Singleton * obj = NULL;
	Singleton();
public:
	static Singleton * getObj() {
		if (obj == NULL)
			obj = new Singleton();

		return obj;
	};
};
```

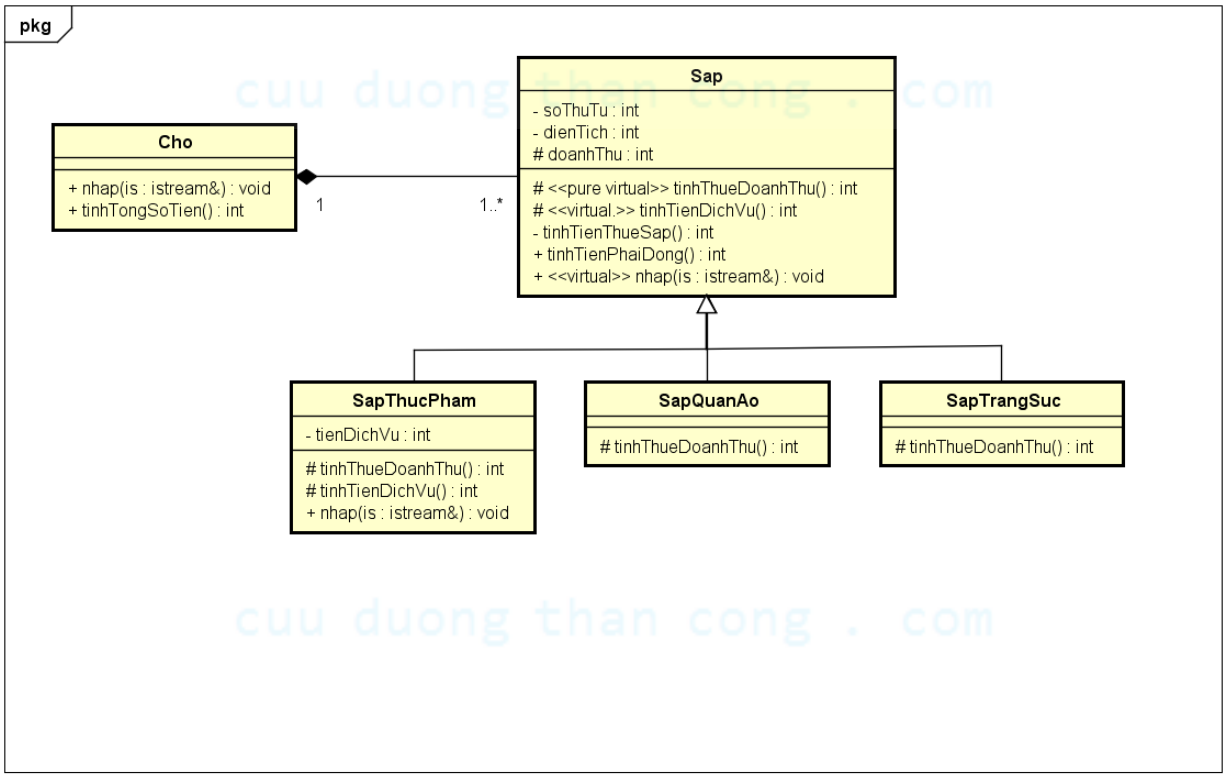
Lớp Singleton chỉ có thể tạo được một thể hiện duy nhất do ta không thể tạo ra lớp Singleton qua việc gọi contructor hay khai báo biến (do constructor là phương thức private). Ta chỉ có thể tạo ra đối tượng Singleton qua việc gọi Singleton::getObj() nhờ đó ta có thể quản lý số lượng đối tượng Singleton được tạo ra. Nếu đã có một đối tượng được tạo ra rồi thì ta sẽ không tạo ra nữa mà trả về đối tượng đó

Câu 2:

```
class BigInteger {
private:
	string number;
public:
	BigInteger(string number = "0") : number(number) {
	}

	bool operator == (const BigInteger & x) const {
		return number == x.number;
	}

	friend istream & operator >> (istream & is, BigInteger & number);
};

istream & operator >> (istream & is, BigInteger & number) {
	return is >> number;
}
```

Câu 3:

Sơ đồ lớp



Cài đặt:

```
#include <iostream>
#include <vector>
```

```
using namespace std;

class Sap {
private:
	const int DON_GIA_SAP = 40000000;

	int soThuTu;
	int dienTich;
	int tinhTienThueSap() {
		return DON_GIA_SAP * dienTich;
	};

protected:
	int doanhThu;
	virtual int tinhThueDoanhThu() = 0;
	virtual int tinhTienDichVu() {
		return 0;
	}
public:
	int tinhTienPhaiDong() {
		return tinhTienThueSap() + tinhTienDichVu() + tinhThueDoanhThu();
	}
	virtual void nhap(istream & is) {
		cout << "Nhap so thu tu: ";
		is >> soThuTu;
		cout << "Nhap dien tich: ";
		is >> dienTich;
		cout << "Nhap doanh thu: ";
		is >> doanhThu;
	}
};

class SapThucPham : public Sap {
private:
	int tienDichVu;
protected:
	int tinhThueDoanhThu() {
		return doanhThu * 5 / 100;
	}
	int tinhTienDichVu() {
		return tienDichVu;
	}
};


class SapQuanAo : public Sap {
protected:
	int tinhThueDoanhThu() {
		return doanhThu * 10 / 100;
	}
};

class SapTrangSuc : public Sap {
protected:
	int tinhThueDoanhThu() {
		if (doanhThu >= 100000000)
			return doanhThu * 30 / 100;
```

```
		else
			return doanhThu * 20 / 100;
	}
};

class Cho {
private:
	vector<Sap *> dsSap;
public:
	void nhap(istream & is) {
		int n;
		cout << "Nhap so luong sap: ";
		is >> n;
		dsSap.resize(n);

		for (int i = 0; i < n; i++) {
			cout << "Nhap loai 1 - sap thuc pham, 2 - sap quan ao, 3 - sap trang
suc: ";

			int loai;
			is >> loai;

			switch (loai) {
			case 1:
				dsSap[i] = new SapThucPham();
				break;
			case 2:
				dsSap[i] = new SapQuanAo();
				break;
			case 3:
				dsSap[i] = new SapTrangSuc();
				break;
			default:
				cout << "Loai sap khong hop le" << endl;
			}

			dsSap[i]->nhap(is);
		}
	};

	int tinhTongSoTien() {
		int total = 0;

		for (int i = 0; i < dsSap.size(); i++)
			total += dsSap[i]->tinhTienPhaiDong();

		return total;
	}
};
```

Đề 2014

Câu 1

a.

Cần cài đặt thêm 3 ông lớn và giá trị mặc định cho hiệu xe

```
Bike() : brand(NULL) {
	set_brand("default");
}

Bike(const Bike & b) : brand(NULL) {
	set_brand(b.brand);
}

virtual ~Bike() {
	if (brand != NULL)
		delete[]brand;
}

Bike & operator = (const Bike & b) {
	set_brand(b.brand);
}

void set_brand(char *brand) {
	if (brand != NULL)
		delete[] brand;

	this->brand = new char[strlen(brand)+1];
	strcpy(this->brand, brand);
	//hoac this->brand = strdup(brand);
}
```

b.

Kết quả là

default:48 default:48

Do hàm display nhận tham số theo kiểu tham chiếu Bike& và move được khai báo hàm ảo nên hàm move của đúng đối tượng EBike sẽ được gọi

c.

Đối tượng là thể hiện của lớp. Một lớp có thể có nhiều đối tượng khác nhau. Mỗi đối tượng đều của 1 lớp sẽ có nhưng phương thức và thuộc tính mà lớp đó quy định nhưng giá trị của thuộc tính có thể khác nhau.

Lớp: Bike, EBike

Đối tượng được lưu trong các biến b1, b2.

Câu 2:

a.

```
class EyeFace : public Face {
private:
	int eyes;
public:
	EyeFace(string sh, int eyes) : Face(sh), eyes(eyes) {
	}
```

```
	virtual void show() {
		Face::show();
		cout << "Eyes: " << eyes << endl;
	}

	EyeFace * clone() {
		return new EyeFace(getShape(), eyes);
	}
};
```

b.

Hàm main không chạy được vì lớp Face là lớp thuần ảo (chưa cài đặt Phương thức clone của IFace) nên không thể tạo đối tượng Face nên ta sẽ cài thêm hàm clone. Tuy nhiên đối tượng fc vẫn không được tạo ra do lớp Face không có constructor mặc định nên ta sẽ cài đặt constructor mặt định.

```
	Face * clone() {
		return new Face(shape);
	}

	Face() :shape("NULL") {

	}
```

Khi đó kết quả xuất ra main là

Shape: Rectangle

Shape: Rectangle

Shape: Rectangle

The same 3 lines?

c.

Do khi clone thì đối tượng mới được tạo ra bằng operator new nhưng lại chưa được delete nên sẽ gây rò rỉ bộ nhớ nên ta sẽ phải delete sau khi dùng

```
void testFace(IFace* fc) {
	IFace* a[3] = { fc, fc->clone(), fc->clone() };
	for (int i = 0; i<3; i++) {
		a[i]->show();
	}
	cout << "The same 3 lines ? ";

	delete a[1];
	delete a[2];
}
```

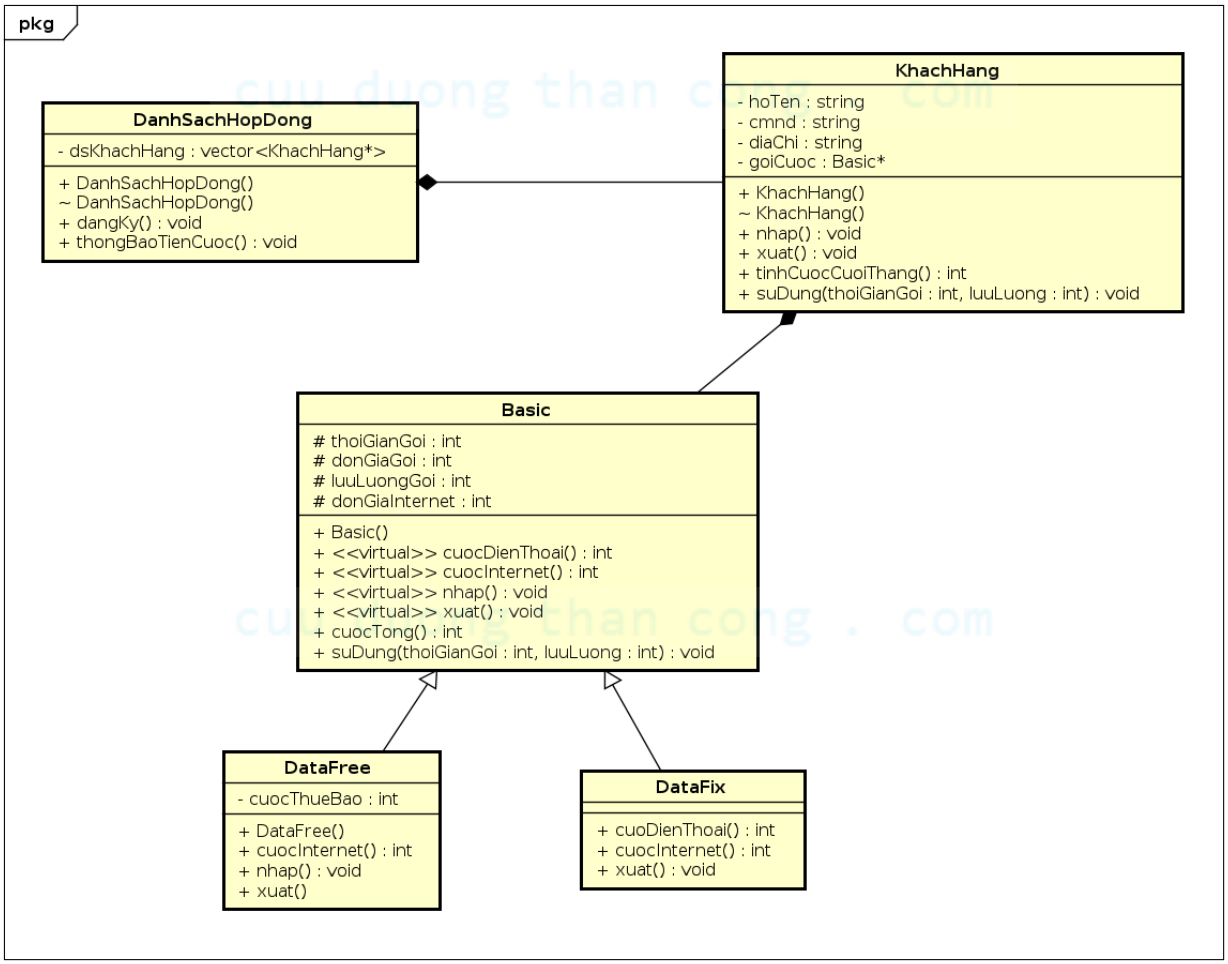
Chỉnh lại lớp EyeFace để thêm một biến static instanceCount để đếm số đối tượng được tạo ra của lớp EyeFace. Khi constructor được gọi thì sẽ tăng biến đó lên 1 và khi destructor được gọi thì sẽ giảm đi 1

```
class EyeFace : public Face {
private:
…
	static int instanceCount;
public:
	EyeFace(string sh, int eyes) : Face(sh), eyes(eyes) {
		instanceCount++;
	}
```

```
        ~EyeFace() {
                instanceCount--;
        }

        static int countNumberOfInstance() {
                return instanceCount;
        }
…
};

int EyeFace::instanceCount = 0;


int main() {
…
        cout << "Number of EyeFace instance: " << EyeFace::countNumberOfInstance() <<
endl;
        return 0;
}
```

Câu 3:

Sơ đồ lớp



Cài đặt

```
class DanhSachHopDong {
	vector<KhachHang*> dsKhachHang;
public:
	void dangKy();
	void thongBaoTienCuoc();
	DanhSachHopDong();
	~DanhSachHopDong();
};

class KhachHang {
	string hoTen;
	string cmnd;
	string diaChi;
	Basic* goiCuoc;
public:
	KhachHang();
	~KhachHang();
	void nhap();
	void xuat();
	int tinhCuocCuoithang();
	void suDung(int, int);
};

class Basic {
protected:
	int thoiGianGoi;
	int donGiaGoi;
	int luuLuong;
	int donGiaInternet;
public:
	virtual int cuocDienThoai();
	virtual int cuocInternet();
	virtual void nhap();
	virtual void xuat();
	int cuocTong();
	void suDung(int, int);
	Basic();
	virtual ~Basic();
};

class DataFix: public Basic {
public:
	int cuocDienThoai();
	int cuocInternet();
	void xuat();
	~DataFix();
};

class DataFree: public Basic {
	int cuocThueBao;
public:
	DataFree();
	int cuocInternet();
	void nhap();
	void xuat();
	~DataFree();
};
```

```
DanhSachHopDong::DanhSachHopDong() {}
DanhSachHopDong::~DanhSachHopDong() {
	for (int i = 0; i < dsKhachHang.size(); i++)
		delete dsKhachHang[i];
}
void DanhSachHopDong::dangKy() {
	int n;
	cout << "Nhap so luong khach hang: ";
	cin >> n;
	dsKhachHang.resize(n);
	for (int i = 0; i < n; i++) {
		cout << "Nhap thong tin khach hang thu " << i + 1 << ":\n";
		dsKhachHang[i] = new KhachHang;
		dsKhachHang[i]->nhap();
	}
}
void DanhSachHopDong::thongBaoTienCuoc() {
	cout << "So luong hop dong: " << dsKhachHang.size() << endl;
	for (int i = 0; i < dsKhachHang.size(); i++) {
		cout << "Thong tin hop dong thu " << i + 1 << ":\n";
		dsKhachHang[i]->xuat();
		cout << endl;
		cout << "Tien cuoc cuoi thang: " << dsKhachHang[i]->tinhCuocCuoithang() <<
endl;
	}
}

KhachHang::KhachHang() {}
KhachHang::~KhachHang() {
	delete goiCuoc;
}
void KhachHang::nhap() {
	cout << "Nhap ho ten: ";
	cin >> hoTen;
	cout << "Nhap cmnd: ";
	cin >> cmnd;
	cout << "Nhap dia chi: ";
	cin >> diaChi;
	int c;
	cout << "Nhap thong tin goi cuoc:\nChon loai goi cuoc (0: Basic, 1: DataFree, 2:
DataFix): ";
	cin >> c;
	if (c == 0)
		goiCuoc = new Basic;
	else if (c == 1)
		goiCuoc = new DataFree;
	else
		goiCuoc = new DataFix;
	goiCuoc->nhap();
}
void KhachHang::xuat() {
	cout << "Ho ten: " << hoTen << endl;
	cout << "CMND: " << cmnd << endl;
	cout << "Dia chi: " << diaChi << endl;
	cout << "Thong tin goi cuoc:\n";
	goiCuoc->xuat();
}
int KhachHang::tinhCuocCuoithang() {
```

```
	return goiCuoc->cuocTong();
}
void KhachHang::suDung(int thoiGianGoi, int luuLuong) {
	goiCuoc->suDung(thoiGianGoi, luuLuong);
}

Basic::Basic() {
	thoiGianGoi = 0;
	donGiaGoi = 1000;
	luuLuong = 0;
	donGiaInternet = 200;
}
Basic::~Basic() {}
int Basic::cuocDienThoai() {
	return thoiGianGoi * donGiaGoi;
}
int Basic::cuocInternet() {
	return luuLuong * donGiaInternet;
}
int Basic::cuocTong() {
	return 1.1 * (cuocInternet() + cuocDienThoai());
}
void Basic::nhap() {}
void Basic::xuat() {
	cout << "Ten goi cuoc: Basic";
}
void Basic::suDung(int _thoiGianGoi, int _luuLuong) {
	thoiGianGoi += _thoiGianGoi;
	luuLuong += _luuLuong;
}

DataFix::~DataFix() {}
int DataFix::cuocDienThoai() {
	return Basic::cuocDienThoai() * 0.9;
}
int DataFix::cuocInternet() {
	return 1000000;
}
void DataFix::xuat() {
	cout << "Ten goi cuoc: DataFix";
}

DataFree::DataFree() {}
DataFree::~DataFree() {}
int DataFree::cuocInternet() {
	return cuocThueBao + (luuLuong <= 0 ? 0: Basic::cuocInternet());
}
void DataFree::nhap() {
	cout << "Nhap cuoc thue bao: ";
	cin >> cuocThueBao;
	cout << "Nhap nguong luu luong mien phi: ";
	cin >> luuLuong;
	luuLuong *= -1;
}
void DataFree::xuat() {
	cout << "Ten goi cuoc: DataFree";
}
```