Solver: Shao Jie

Email Address: yews0012@e.ntu.edu.sg

1. (a)

Object - An entity that contains both the attribute that describes the state of a real-world object and the actions that are associated with the real-world object.

(b)

- State of an object
- Attributes
- Behavior of an object

(c)

- Class name should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
- Method name should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
- Variable name should start with lowercase letter e.g. firstName, orderNumber etc.
- Package name should be in lowercase letter e.g. java, lang, sql, util etc.

(d)

(i) ClassP's moveIt(o:Oject): void
(ii) Error, instance of ClassBP cannot be cast to ClassB
(iii) ClassBP's feedIt(v: ClassV, z: ClassM): void
(iv) ClassM's feedIt(o:Object): void
(v) ClassM's feedIt(o:Object): void
(vi) ClassP's moveIt(o: Object): void

2.
```
class HourlySaleApp{
    public static void main(String arg[]){
        Sale normalSale = new Sale("Samsung Galaxuy Note 7",600.00);
        DiscountSale discountSale = new DiscountSale("Samsung Galaxuy Note
7",600.00, 0.5);
        System.out.println(normalSale.lessThan(discountSale));
    }
}

class Sale{
    private String name;
    private double price;
    public Sale(String name, double price){
        this.name = name;
        this.price = price;
    }
    public double getPrice(){
        return price;
    }
    public boolean lessThan(Sale otherSale){
        if(price<otherSale.getPrice()){
            return true;
        }
        return false;
    }
    public double bill(){
        System.out.println("price: "+price);
    }
}

class DiscountSale extends Sale{
    private double discount;
    public DiscountSale(String n, double p, double d){
        super(n,p);
        this. discount=d;
    }
    public double bill(){
        System.out.println("price: "+(getPrice()*discount));
    }
}
```

3. (a) vector3d.h

```
#include "vector2d.h"
#include "fraction.h"
class Vector3D : public Vector2D
{
    protected Fraction z;
    public:
            Vector3D(Fraction x, Fraction y, Fraction in_z);
            Vector3D(int nx, int dx, int ny, int dy, int nz, int ny) ;
            void print();
            Vector2D& operator+(Vector2D &v);
};
```

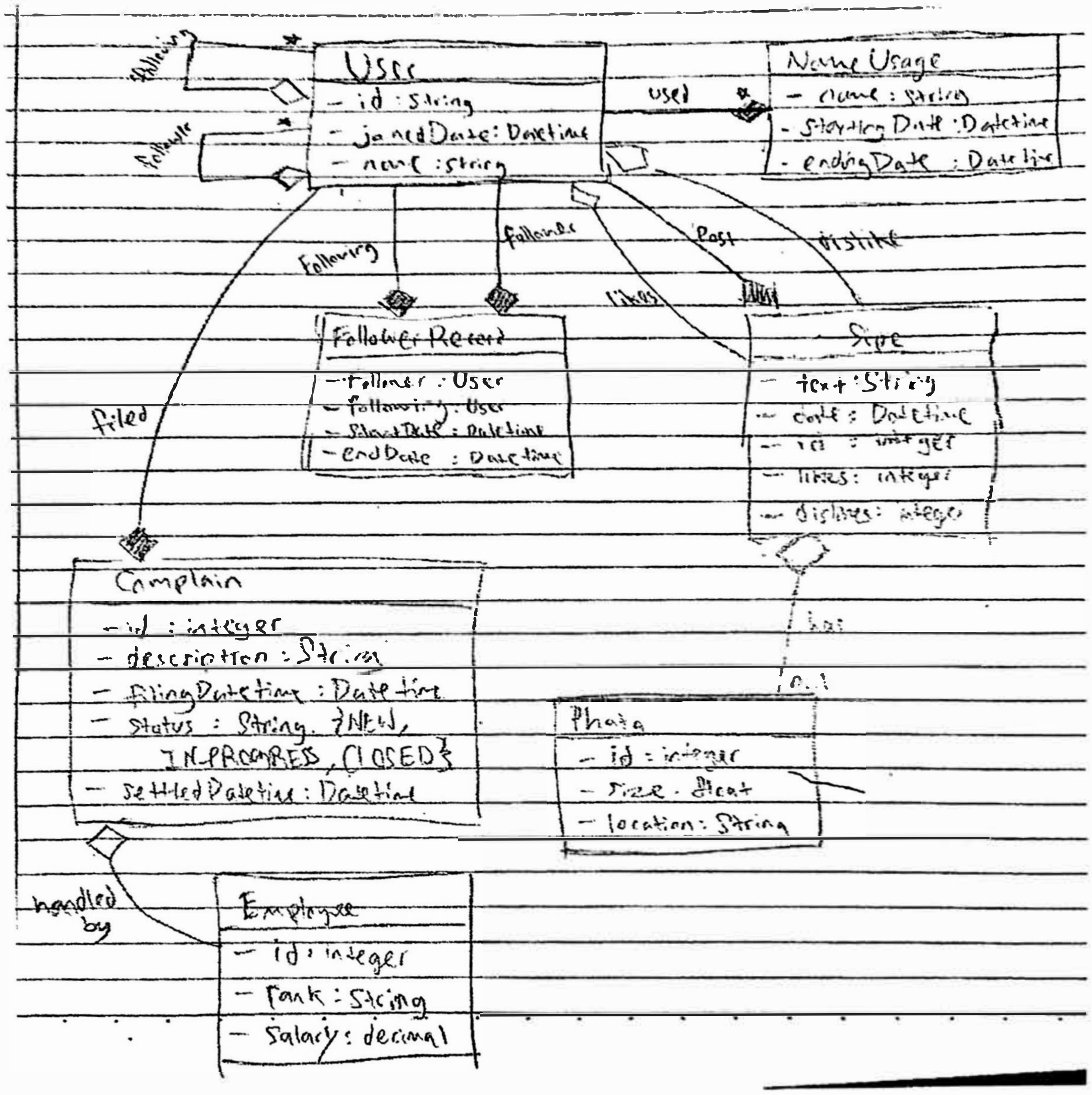
(ii)

```
#include "vector3d.h"

Vector3D::Vector3D(Fraction x, Fraction y, Fraction in_z) : Vector2D (x,y){
        z=in_z;
 }
Vector3D::Vector3D(int nx, int dx, int ny, int dy, int nz, int ny) : Vector2D (nx,dx,ny,dy) {
        z = new Fraction(nz,ny);
}
Vector3D::print(){
        cout << Vector2D::x <<"I "<<Vector2D::y<<"j "<<z<< "k"<<endl;
}
Vector3D:: operator+( Vector2D &v){

        Vector3D &a = dynamic_cast<Vector3D&>(v);
        if (a == NULL){
                return *this;
        }
        return new Vector3D(a.Vector2D::x+ Vector2D::x, a.Vector2D::y+ Vector2D::y, a.z+ z);
}
```

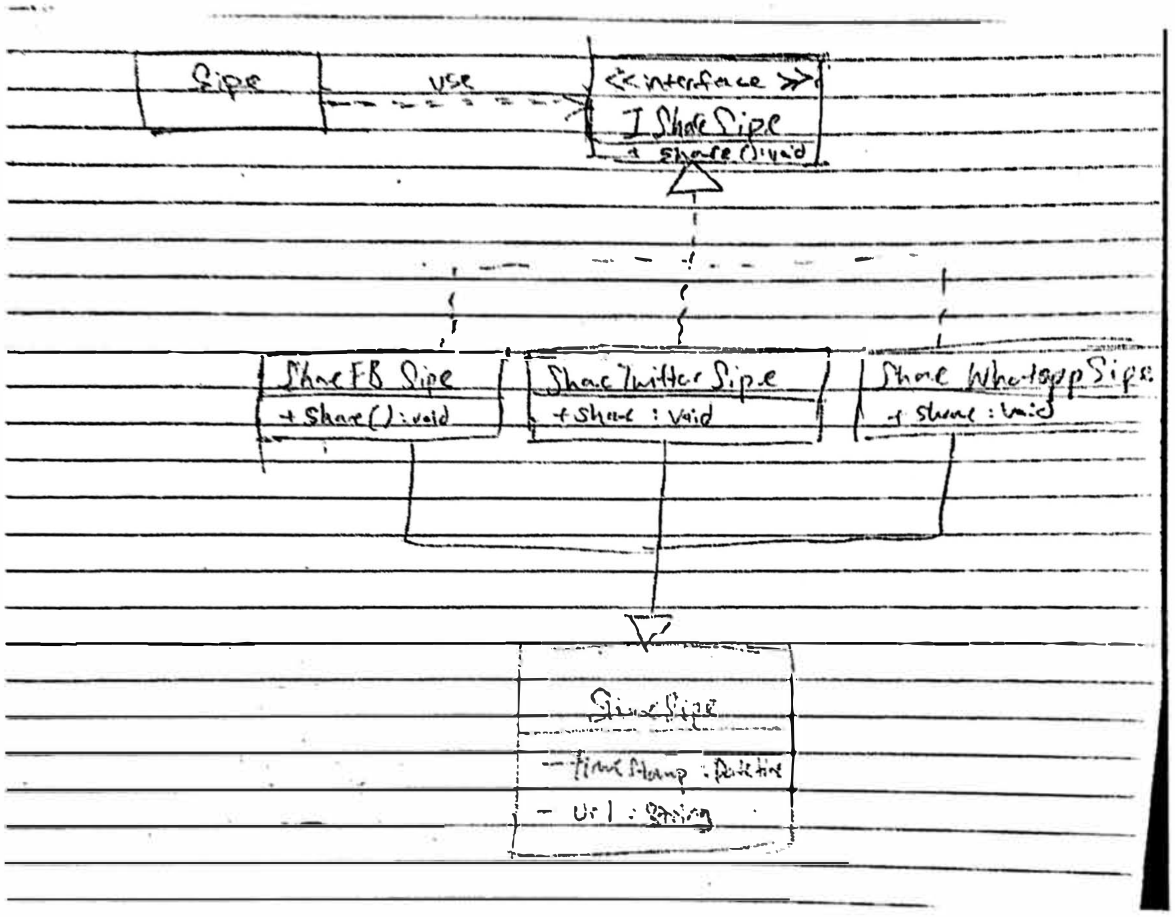
(b)

```
class Local{
        public boolean emit(String msg, String type){
                if(type!=null){
                        Emitter em = new Emitter(type, this);
                        boolean status = false;
                        for(int count=0; count <10;count++){
                                status = em.connect();
                                if(status){
                                        em.send();  break;
                                }
                        }
                        return status;
                }else{
                        displayError();
                        return false;
                }
        }
        public String getAddress(){
                return 'address';
        }
        public void displayError(){
                System.out.println("error");
        }
}
class Emitter{
        private String type;
        private Local local;
        private String addr;
        public Emitter(String type, Local local){
                this.type= type;
                this.local = local;
        }
        public boolean connect(){
                return true;
        }
        public void send(){
                addr = local.getAddress();
        }
}
```

4. (a)


User
- id : String
- joinedDate : Datetime
- name : String
following
follower
*
*
used
*
Name Usage
- name : String
- startingDate : Datetime
- endingDate : Datetime
Following
follower
Post
dislike
likes
FollowerRecord
- follower : User
- following : User
- startDate : Datetime
- endDate : Datetime
Post
- text : String
- date : Datetime
- id : integer
- likes : integer
- dislikes : integer
filed
has
0..1
Complain
- id : integer
- description : String
- filingDatetime : Datetime
- status : String {NEW, INPROGRESS, CLOSED}
- settledDatetime : Datetime
Photo
- id : integer
- size : float
- location : String
handled by
Employee
- id : integer
- rank : String
- salary : decimal

(b)



ShareFBSipe, ShareTwitterSipe and ShareWhatsappSipe class contain corresponding implementation needed for sharing Sipe on Facebook, Twitter and Whatsapp. ShareSipe class contain the common attributes and functions that are required for sharing Sipe. This class allow reusing of common attributes and functions when sharing Sipe. As shown in the above drawing, ShareFBSipe, ShareTwitterSipe and ShareWhatsappSipe class inherited ShareSipe for the **reusability** and at the same time it avoid duplication of code thus it will have better **maintainability**. Modification would cause less inconsistency.

An Interface IShareSipe is used and implemented by different sharing strategy; ShareFBSipe, ShareTwitterSipe and ShareWhatsappSipe class. This helps to provide common interface so that new sharing strategy can be added by implementing the interface. Thus, provide a better **extensibility**. Interface also enforce all the implemented classes to declare the function required. Thus, changes made in the implementation will not affect classes that are using them as long as the interface remains the same.