C++ Programming Polymorphism Homework 2

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher PhD from Simon Fraser University - Canada Bachelor / Msc from Cairo University - Egypt Ex-(Software Engineer / ICPC World Finalist)



Homework 1: Extended Company Payroll

- We need to extend the previous company payroll
 - o In practice, we may need to <u>verify invoices</u>. Sometimes an:
 - invoice is out-of-date relative to some deals with suppliers
 - Computed taxes doesn't utilize some advantages in the country
 - So there are several validation rules
 - In future, more rules might be added (we need generic code)
 - You don't need to care about validation rules logic
 - A Validator Group consists of a subset the available validation rules
 - E.g. Mandatory-Validator has a very few validation rules to be fast
 - Based on configuration, a specific Validator Group is in use
 - An invoice must pass all validation rules in a validator group

Homework 2: Expedia Travel Site

- In expedia a user has several itineraries, each itinerary consists of several reservations as following
 - o or more flights, hotels, cars, etc. E.g. 4 flights, 2 hotels and 2 cars.
 - Consider only flights (one way no transit) and hotels (but create extensible design).
- An itinerary item (e.g. a flight) has:
 - Relevant info: E.g. Hotel cost: total nights x price per night
 - Itinerary cost is the **sum** of all internal items costs.
- Design the set of classes that help developing the system + UML
 - Task scope: Only **reservation** classes + relevant concerns

- <u>Craigslist</u> website (classified-ads) is adding new feature for customers payment.
 - There are a lot of API to use. Each API has fees in some style
 - E.g. 1 dollar per 25 payments
 - The rules might change from time to time
 - New APIs with new fees might be available on web
 - All APIs provide similar functionalities, but different interfaces
 - At the moment, developers want to support to payment methods: PayPal and Stripe
 - In future, more could be added if they are cheaper
 - The site code to pay something shouldn't depend on specific API.
 - Otherwise, a lot of code has to be changed with every a change in the selected API

```
5⊖ class PayPalCreditCard {
 6 public:
        string name;
       string address;
       string id;
       string expire date;
10
11
       int ccv:
12 };
13
140 class PayPalOnlinePaymentAPI {
15 public:
       void SetCardInfo(const PayPalCreditCard* const card) {
169
17
       bool MakePayment(double money) {
189
19
            return true;
20
21 };
```

- Developers investigated
 Paypal provided payment API
 - Seems we create object of the API
 - Set info through the PayPalCreditCard
 - Then call MakePayment
 - Bool to indicate sucess

```
23@ class StripeUserInfo {
24 public:
       string name;
       string address;
26
27 };
28
29@ class StripeCardInfo {
   public:
31
        string id:
        string expire date;
33 };
34
35@ class StripePaymentAPI {
36 public:
        bool static WithDrawMoney(StripeUserInfo user,
38
                                   StripeCardInfo card,
39
                                   double money) {
40
            return true:
```

- For Stripe payment API
 - Seems we use API in static way
 - Also 2 seperate objects to fill info
 - Ccv info not included
 - Set info through the PayPalCreditCard
 - Then call WithdrawMoney
 - Bool to indicate sucess

```
440 class TransactionInfo {
45 public:
       double required money amount;
       string name;
       string address;
       string id;
       string expire date;
50
51
       int ccv;
52 };
53
540 class Craigslist {
       public:
55
       bool Pay(TransactionInfo) {
            //TODO: generic (no nothing about Paypal)
```

- Let's develop our site
- We want to implement this Pay function
- It should be generic
 - E.g. No dependency on PayPal
- Add whatever to do so
 - You can't change API
 - It is not our code

Homework 4: Extended Payment Services

```
44⊖ class SquarePaymentAPI {
45 public:
46⊖ bool static WithDrawMoney(string JsonQuery) {
```

```
{
    "card_info" : {
        "CCV" : 333,
        "DATE" : "09-2021",
        "ID" : "11-22-33-44"
    },
    "money" : 20.500000,
    "user_info" : ["mostafa", "canada"]
}
```

- New payment method is cheaper
- The interface is simpler
 - The request info is represented using a JSON query
 - See example
- Extend your code for the new request
 - Expected minimal changes
 - Craigslist part is not changed

Homework 5: Another Company Payroll

- Recall we have Employee and Invoice which are Payable
 - But implement this separately for simplicity.
 - E.g. no types of employees. Employee has name and salary
- Payable class should be Cloneable, Printable and Comparable
- CompanyPayroll, which is collection of Payables, should be Printable and Sortable
 - Sortable means we can call function to order the collection of items
 - To sort the collection, each type should be grouped together
 - Within each group, should be compared based on comparable property
 - E.g. for employee: sort by salary, then by name
 - E.g. employees printed first, then invoices after them

Homework 5: Another Company Payroll

```
CompanyPayroll payroll:
payroll.AddPayable(Employee(50, "mostafa"));
payroll.AddPayable(Invoice(200));
payroll.AddPayable(Employee(10, "ziad"));
payroll.AddPayable(Invoice(100));
payroll.AddPayable(Employee(30, "belal"));
payroll.AddPayable(Invoice(300));
payroll.OrderItems();
payroll.Print(); // Notice ordered
Employee ziad has salary 10
Employee belal has salary 30
Employee mostafa has salary 50
Invoice cost 100
Invoice cost 200
Invoice cost 300
Total to be paid: 690
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."