به نام خدا

فاز ۲و۳ پروژه پایانی

پروژه مکانیک

صالح اصلانی

در این دو فاز هدف ما اضافه کردن ارث بری و پلیمرفیزم و اضافه کردن اکسپشن و وکتور میباشد.

برای استفاده از ارث بری کلاسی به نام tools را اضافه میکنیم.

```
#ifndef TOOLS_H
#define TOOLS_H

#include <string>
#include <stdexcept>

class Tools
{
    public:
        int price;
        virtual std::string get_info() = 0;
};

#endif
```

این کلاس را بصورت ابسترکت تعریف میکنیم زیرا میدانیم که از این کلاس قرار نیست شی ساخته شود و هدف ما از ایجاد این کلاس این است که کلاس های اینه بقل و موتور از ان ارث ببرند.

```
enum mirror_quality {italian = 1 , japonese , iranian};
enum mirror_type {straight = 1 , curved};

class Wingmirror: public Tools
{
    public:

        Wingmirror(mirror_quality a, mirror_type p);
```

```
        void set_mirror_quality(mirror_quality a);
        void set_mirror_type(mirror_type p);


        void set_price(int);
        int get_price() const;
        std::string get_info();

    private:
        mirror_quality quality;
        mirror_type mir_type;
        int price=5;
};
```

کلاس اینه بقل از کلاس tools ارث برده است

برای اینکه کلاس اینه بقل ابسترکت نباشد حتما لازم است که تابع get_info برای ان پیاده سازی شده باشد که این کار را انجام دادم.

```
enum engine_strengh {weak = 1 , medium , high};
enum engine_quality {japonese_e = 1 , italian_e};

class Engine: public Tools
{
    public:

        Engine(engine_strengh a, engine_quality p);

        void set_engine_strength(engine_strengh a);
        void set_engine_quality(engine_quality p);
        std::string get_info();

        void set_price(int);
        int get_price() const;


    private:
        engine_strengh strength;
        engine_quality enj_quality;

};
```

کلاس موتور نیز از کلاس tools ارث میبرد

مانند کلاس اینه بقل پیاده سازی تابع get_info اینجا نیز اجباریست.

سپس برای اینکه بتوانیم تمام اشیایی که تولید میشوند را در کلاس مکانیک ذخیره کنیم یک وکتور از نوع اشاره گر به کلاس tools تعرف میکنیم و به عنوان ممبر برای کلاس مکانیک در نظر میگیریم

```
class Mechanic
{
```

```
    public:

        Mechanic(std::string ="user");

        void set_name(std::string &);
        std::string get_name() const;

        void increase_money(int);
        void decrease_money(int);
        int get_money() const;

        std::vector<Tools*> mytools;
        void print_stuff();

    private:
        std::string name;
        int money;
};
```

در نهایت برای استفاده از اکسپشن ها نیز مفهموی را تعریف میکنیم که کاربر نتواند به یکباره مقداری بیش از ۱۰۰۰ دلار به حساب خود واریز کند و اگر عدد بیشتر از ۱۰۰۰ باشد این عدد با استفاده از فرایند اکسپشن به ۹۹۹ تغییییر میکند.

```
void Mechanic::increase_money(int m)
{
    if(m>1000)
    {
        throw out_of_range("you cant add this much money to your at this point");
    }
    else{
    this->money += m;
    }
}
```

```
Mechanic a("saleh");
```

```
    try{
        a.increase_money(2000);
        }
    catch(out_of_range &s)
    {
        cout << s.what() << " your wallet will be set to 999$" << endl;
        a.increase_money(999);
    }
```

پایان فاز ۲و۳