Aluno: Gabriel Nunes Crispino

Fica registrado nesse documento os requisitos antes não completados na atividade de herança. Agora completos.

Requisitos herança:

- Diagrama de classes (obrigatório salvar também o png do diagrama no gitHub) - OK
- Herança pública

  - Requisito cumprido, visto documento mandado previamente.
- Construtor de cópia, e sobrecargas dos operadores de atribuição (=) e << (cout << base) para a classe base e derivada

  - Requisito cumprido, visto documento mandado previamente.
- Usar Protected acessando diretamente os atributos na classe derivada

  Requisito cumprido, visto documento mandado previamente.
- Alocação dinâmica de memória na classe base e derivada

  - Na classe base, há um array para objetos da classe Data, que salva por alocação dinâmica de memória as datas de manutenção do dispositivo:

  Device.h:

```
Data *datasmanutencao; //array que guarda as datas em que o dispositivo sofreu manutenção.
int ndatasmanutencao;//número de datas de manutenção.
```

Device.cpp

```
void Device::setDatasManutencao(const Data &d){
    Data * aux;
    if (this->ndatasmanutencao == 0){
        this->datasmanutencao = new Data(d);
        this->ndatasmanutencao++;
    }
    else{
        aux = new Data[this->ndatasmanutencao];
        int i;

        for (i = 0;i < this->ndatasmanutencao;i++)
            aux[i] = this->datasmanutencao[i];

        if (this->ndatasmanutencao == 1)
            delete this->datasmanutencao;
        else
            delete [] this->datasmanutencao;

        this->datasmanutencao = new Data[this->ndatasmanutencao];

        for (i = 0;i < this->ndatasmanutencao;i++)
            this->datasmanutencao[i] = aux[i];

        this->datasmanutencao[this->ndatasmanutencao++] = Data(d);
        delete [] aux;
    }
```

- -Na classe derivada, há um ponteiro para usuários cuja alocação é feita durante o programa na função abaixo:

  Caixa_Eletronico.h

```
Usuario *userpadrao; //array de usuários registrados no caixa
```

  Caixa_Eletronico.cpp

```
void Caixa_Eletronico::setUsuario(const Usuario &u){
    if (this->nusuarios == 0){
        //Se o numero de usuarios for igual a zero, é alocada memória para o primeiro usuário
        this->userpadrao = new Usuario();
        this->userpadrao[0] = u;
        this->nusuarios++;
    }
    else{
        //Se não for igual a zero, faz a realocação para mais um elemento.
        Usuario *aux = new Usuario[nusuarios];

            for (int i = 0;i < nusuarios;i++)
                aux[i] = this->userpadrao[i];

            delete this->userpadrao;

            this->userpadrao = new Usuario[++this->nusuarios];

            for (int i = 0;i < nusuarios-1;i++)
                this->userpadrao[i] = aux[i];

            this->userpadrao[nusuarios-1] = u;

            delete [] aux;
    }
}
```

- Sobrescrita de método: chamar dentro do método da classe derivada o método correspondente da classe base usando ::

  - Dentro da função de sobrecarga do operador “<<” da classe Caixa_Eletronico, é chamada a sobrecarga da função Device:

  Caixa_Eletronico.cpp:

```
ostream &operator << (ostream &output,const Caixa_Eletronico &c)
{
    output<<endl;
    output<<"\n-- Informacoes gerais do caixa eletronico: \n";
    cout<<(Device)c;
    output<<"\nDinheiro disponivel no caixa: R$"<<c.dinheiro<<".";
    output<<endl<<"Numero de usuarios cadastrados: "<<c.nusuarios<<" .";
    output<<endl<<"Numero total de contas cadastradas: "<<c.ntotalcontas<<" .";

    output<<endl;

    return output;
}
```

- No main: criar um ponteiro da classe base para alocar memória para a classe derivada e chamar os vários métodos implementados

  No main, é criado um ponteiro para a classe base Device chamado “ptrdispositivo”, que aponta para um objeto alocado dinamicamente da classe Caixa_Eletronico, que é uma cópia do objeto “c”, que é a instância principal da classe Caixa_Eletronico usada no programa principal.

  Desse ponteiro, são chamadas as funções para a classe base utilizadas na função main.

main.cpp

```
Device *ptrdispositivo = new Caixa_Eletronico(c);
```

```
case 3:
    system("cls");
    Caixa_Eletronico::mostrarData();
    cout<<c;
    getch();
    cout<<endl<<"Deseja visualizar as datas de manutencao do caixa(S ou N)?";
    cin >> r;
    r = toupper(r);
    if (r == 'S')// se o usuário desejar, imprime as datas de manutenção do dispsitivo.
        ptrdispositivo->imprimeDatasManutencao();
    else if (r != 'N')
        cout<<endl<<"Opcao invalida!";
    break;
```

```
case 2:
    system("cls");
    cout<<endl<<"Digite a senha de administrador do dispositivo para entrar no menu de manutencao: ";
    cin >> senha;
    if (ptrdispositivo->verificaSenha(senha)){
        cout<<endl<<"--Menu de manutencao--";
        getch();
        cout<<endl<<"Digite a data da manutencao:";
        cout<<endl<<"- Dia: ";
        cin >>dia;
        cout<<endl<<"- Mes: ";
        cin >> mes;
        cout<<endl<<"- Ano: ";
        cin >> ano;

        ptrdispositivo->setDatasManutencao(Data(dia,mes,ano));
    }
    else{
        cout<<endl<<"Senha incorreta!";
        getch();
    }
    break;
```

PS: Alguns requisitos foram cumpridos quando a classe base ainda era “Banco”, e não “Device”. A modelagem de classes mudou, mas esses requisitos continuaram sendo cumpridos com essa mudança.