Programação Orientada a Objetos
Projeto Orientado a Objetos
Valor: 70 pts

1. Orientações Gerais

O trabalho pode ser realizado em grupos de até quatro alunos e deverá ser entregue no moodle de forma incremental a cada semana, sempre no domingo até às 23:59. É permitida a discussão de estratégias com os colegas, porém a implementação somente deve ser feita pelos integrantes do grupo. Cópias de programas (ainda que parciais) não serão aceitas. Entregas com atraso poderão ser aceitas, porem poderão ser penalizadas com perda de pontos pelo grupo. Deve ser postado apenas um arquivo compactado contendo todos os entregáveis por grupo. Apenas um aluno deve realizar a entrega informando o nome dos integrantes do grupo. Os grupos não devem ser alterados ao longo do semestre.

2. Objetivo

Neste trabalho prático desenvolveremos um modelo de classes e implementaremos um sistema de gestão para uma distribuidora de energia elétrica. O projeto será desenvolvido em sprints semanais, que envolverão a modelagem das classes para atender ao escopo da sprint e também a implementação dessas classes em C++. O código-fonte das classes deverá ser testado através da chamada de métodos das mesmas.

A dinâmica de trabalho se dará da seguinte forma:

- A sprint se inicia em uma sexta-feira, na qual serão apresentados os requisitos a serem atendidos;
- A modelagem das classes em linguagem UML deverá ser entregue via moodle até o domingo subsequente;
- Na quinta-feira da semana seguinte, a implementação das classes será iniciada na aula de laboratório.
- O código-fonte gerado deverá ser enviado via moodle também até o domingo subsequente.

A pontuação se dará da seguinte forma:

- 10 pts para a entrega de cada sprint, sendo 3 pts para a modelagem e 7 pts para o código-fonte.

2.1. Sprint 1

Nesta sprint vamos iniciar o desenvolvimento do Módulo Comercial, que deverá atender aos seguintes requisitos:

- Permitir o cadastro de clientes.
- Permitir o cadastro de unidades consumidoras, vinculadas aos clientes. Cada unidade consumidora pode ser residencial, comercial, industrial, iluminação pública.
- Permitir o cadastro de faturas mensais para cada UC, contendo energia consumida (kWh) e valor da fatura (R$).
- Realizar o registro de pagamentos de fatura pelos clientes.
- Realizar o cálculo de juros diário sobre atrasos de faturas, a partir de uma taxa mensal fixa.
- Gerar uma lista de clientes inadimplentes com suas devidas faturas em atraso.
- Clientes pessoa física podem ter UC's dos tipos residencial e comercial. Enquanto que clientes pessoa jurídica podem ter UC's dos tipos comercial, industrial e iluminação pública. CPF e CNPJ devem ser validados.

2.2. Sprint 2

Este módulo deverá permitir o registro de um serviço de campo a ser executado por um funcionário da distribuidora, que compreendem os serviços de *medição* (leitura), *desligamento por inadimplência*, *desligamento por encerramento*, *ligação nova*, *religação por pagamento* e *troca de medidor*.

Cada serviço é programado para ser executado em uma Unidade Consumidora, por um funcionário, em uma data e sequência no dia. Um funcionário somente poderá executar no máximo 8 serviços por dia.

Após a execução do serviço, o funcionário da distribuidora deverá registrar a data e hora da execução do mesmo.

Em um dia de trabalho, o funcionário extrai do sistema um relatório com todos os serviços planejados para aquele dia no início da sua jornada de trabalho.

É obrigatório o uso de herança na implementação das classes que modelam os tipos de serviço.

2.3. Sprint 3

Nesta semana vamos incluir mais alguns atributos em nossas classes e implementar novas regras de negócio.

- A classe que modela o cliente deve contar com os seguintes atributos: nome, endereço e telefone de contato.
- O endereço deverá ser modelado como herança a partir de uma classe *LocalizacaoGeografica*, que possui apenas os atributos latitude e longitude. Deve conter: logradouro, número, bairro, complemento, cep, cidade, estado.
- Unidades Consumidoras também devem ter um endereço associado.
- Toda vez que um serviço de medição (leitura) for realizado, deve ser gerada automaticamente uma fatura para aquela UC. O valor lido corresponderá ao valor contínuo medido pelo leiturista no medidor. Ou seja, para se obter a quantidade de energia consumida em kWh no período deve-se subtrair do valor lido a medição anterior. Esse valor será utilizado para o cálculo da fatura mensal.
- Todas as datas do sistema deverão utilizar a classe *Data* (em anexo).
- O cálculo de juros deverá agora comparar a data atual (hoje) com a data de vencimento da fatura, utilizando o método *diffData()*. Com base no cálculo de número de dias de um mês típico de 30 dias, o valor dos juros em R$ deverá ser calculado.

....

Bom trabalho!