Universidade Federal de Campina Grande

Relatório de Projeto

Disciplina: Laboratório de Programação 2
Período: 2018.1

Equipe: *Henry Maldiney de Lira Nóbrega Filho, Igor Santa Ritta Seabra, Wesley Roseno Saraiva*

Introdução

O grupo iniciou fazendo o planejamento básico do design do projeto. Seria feita uma classe fachada que iria interagir com o sistema apenas chamando e retornando seus métodos, para diminuir o acoplamento. As classes iriam interagir apenas com aquelas que estivessem contidas nelas por composição, sendo responsáveis por chamar seus construtores e gerencia-las nas coleções. Cada membro do grupo ficaria responsável por uma parte das classes e pelos testes de unidade, para evitar conflitos no compartilhamento do código, mas poderia fazer alterações nas outras caso fosse necessário. As funcionalidades foram planejadas com base na especificação do projeto e nos testes de aceitação fornecidos pelos professores; procuramos também escrever código eficiente de um ponto de vista computacional.

O projeto pedia a criação de um sistema que gerenciasse listas de compras.

Caso 1

O caso de uso 1 pedia que o sistema fosse capaz de armazenar e gerenciar itens de compra. Haveriam diferentes tipos de item, com propriedades em comum mas alguns atributos diferentes. Utilizamos uma classe Produto abstrata como base e herança para os diferentes tipos de produto, para o polimorfismo facilitar a gerência e reuso de código, além da fácil adição de novos tipos caso fosse necessário. Os atributos de Produto são abstratos e acessados através de métodos get e set (que verificam o valor recebido e validam caso necessário) para evitar atribuições indevidas. Escolhemos usar um mapa para armazenar os produtos, pois seriam identificados unicamente e facilmente recuperados por seu identificador único (valor inteiro gerenciado pelo sistema).

Caso 2

O caso de uso 2 pedia que o sistema exibisse representações em texto dos produtos cadastrados de diferentes formas. Para isso, criamos métodos na fachada e no sistema que percorrem a coleção de produtos e geram o texto de acordo com a entrada. Como algumas das formas de representação pediam mais de um produto, escolhemos gerar listas temporárias e ordená-las com os métodos padrão de ordenação de listas da biblioteca de Java porém com nossos próprios métodos comparadores.

Caso 3

O caso de uso 3 pedia que o sistema fosse capaz de armazenar e gerenciar listas de compra. As listas de compra seriam identificadas pelo seu nome (código) e armazenariam uma lista de itens (que deveriam já estar armazenados no sistema) e uma quantidade para cada um. Para isso, criamos uma classe ListaDeCompras, que seria gerenciada em um mapa pelo sistema, e uma classe Compra, que seria gerenciada em uma lista em ListaDeCompras. A classe Compra foi escolhida pois poderia armazenar uma quantidade e uma referência para o produto, evitando que ListaDeCompras tivesse acesso direto a Produto mas facilitando que características de Produto fossem acessadas sem necessidade de voltar ao sistema. A especificação pedia também que a representação textual da lista de compras tivesse suas compras ordenadas em uma

ordem específica; para isso, utilizamos os métodos padrão de ordenação de listas com nossos próprios métodos comparadores.

Caso 4

O caso de uso 4 pedia que o sistema exibisse representações em texto das listas de compras de diferentes formas. Para isso, criamos métodos na fachada e no sistema que percorrem a coleção de listas e verificam quais atendem às condições pedidas; as que atendessem eram então ordenadas e sua representação textual retornada.

Caso 5

O caso de uso 5 pedia que o sistema gerasse automaticamente listas de compras de acordo com critérios pré-estabelecidos. Alguns desses métodos necessitavam que armazenássemos a ordem na qual as listas foram criadas, algo que o sistema previamente não fazia. Para isso adicionamos um atributo identificador numérico à ListaDeCompras. Para o caso em que a nova lista deve ter os produtos que mais aparecem em listas já existentes, foi necessária a criação de uma nova classe, a Tupla, que armazena dois valores inteiros. Essa classe foi usada de diferentes formas, tanto para obter das listas de compras o identificador único do produto e sua quantidade, quanto a quantidade de listas em que o produto aparece juntamente de sua quantidade total, para facilitar o cálculo das médias necessárias.

Caso 6

O caso de uso 6 pedia também a geração automática de listas de compras, porém agora as novas listas teriam uma única lista como base e seriam divididas de acordo com os estabelecimentos disponíveis para os produtos da lista, e ordenadas de acordo com o menor preço total somando os produtos disponíveis naquele estabelecimento, para sugestão de um melhor estabelecimento para aquela lista. Para isso, foi criado um método em ListaDeCompras que realiza essa função percorrendo sua própria lista de produtos e retornando as novas listas, para que não fosse necessário que o sistema interagisse com os produtos das listas.

Caso 7

O caso de uso 7 pedia que o programa fosse capaz de salvar os dados localmente para recuperação posterior. Para isso, todas as classes do projeto que são armazenadas em coleções e a própria classe Sistema receberam a implementação de Serializable e foi criada uma classe nova, SistemaIO, que possui métodos estáticos para escrever e ler o sistema de um arquivo. Na fachada foram implementados os métodos que enviam o sistema a essa classe e o recuperam a partir dela, sendo os únicos métodos da fachada que não chamam métodos de Sistema justamente pela sua necessidade de enviar toda o objeto a um método ou sobrescreve-lo por uma nova instância.

Repositório do projeto no GitHub