UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
IMD0030 – LINGUAGEM DE PROGRAMAÇÃO I

Laboratório 3

Herança, Classes Abstratas e Manipulação de arquivos

Objetivo

O objetivo desta lista de exercícios é praticar os conceitos discutidos em sala de aula sobre a Programação Orientada a Objetos (POO) em linguagem C++, em particular a implementação de herança, classes abstratas e a manipulação de arquivos.

Orientações gerais

Você deverá observar as seguintes observações gerais na implementação deste exercício:

1) Deverão ser utilizados **estritamente** recursos da linguagem C++.
2) Durante a compilação do seu código fonte, você deverá habilitar a exibição de mensagens de aviso (*warnings*), pois elas podem dar indícios de que o programa potencialmente possui problemas em sua implementação que podem se manifestar durante a sua execução.
3) Aplique boas práticas de programação. Codifique o programa de maneira legível (com indentação de código fonte, nomes consistentes, etc.) e documente-o adequadamente na forma de comentários. Anote ainda o código fonte para dar suporte à geração automática de documentação utilizando a ferramenta Doxygen (http://www.doxygen.org/). Consulte o documento extra disponibilizado na Turma Virtual do SIGAA com algumas instruções acerca do padrão de documentação e uso do Doxygen.
4) Busque desenvolver o seu programa com qualidade, garantindo que ele funcione de forma correta e eficiente. Pense também nas possíveis entradas que poderão ser utilizadas para testar apropriadamente o seu programa e trate adequadamente possíveis entradas consideradas inválidas.
5) Lembre-se de aplicar boas práticas de modularização, em termos da implementação de diferentes funções e separação entre arquivos cabeçalho (`.h`) e corpo (`.cpp`).
6) A fim de auxiliar a compilação do seu projeto, construa **obrigatoriamente** um `Makefile` que faça uso da estrutura de diretórios já discutida em aula.

Autoria e política de colaboração

Este trabalho deverá ser realizado **individualmente**. O trabalho em cooperação entre estudantes da turma é estimulado, sendo admissível a discussão de ideias e estratégias. Contudo, tal interação não deve ser entendida como permissão para utilização de (parte de) código fonte de colegas, o que pode caracterizar situação de plágio. Trabalhos copiados em todo ou em parte de outros colegas ou da Internet serão sumariamente rejeitados e receberão nota zero.

Você deverá utilizar o sistema de controle de versões Git no desenvolvimento. Ao final, todos os arquivos de código fonte do repositório Git local deverão estar unificados em um repositório remoto Git hospedado em algum serviço da Internet, a exemplo do GitHub, Bitbucket, Gitlab ou outro de sua preferência. A fim de garantir a boa manutenção de seu repositório, configure corretamente o arquivo `.gitignore` em seu repositório Git.

Entrega

Você deverá submeter um único arquivo compactado no formato `.zip` contendo todos os códigos fonte resultantes da implementação deste exercício, sem erros de compilação e devidamente testados e documentados, **até as 23:59h do dia 20 de outubro de 2017** através da opção *Tarefas* na Turma Virtual do SIGAA. Você deverá ainda informar, no campo *Comentários* do formulário de submissão da tarefa, o endereço do repositório Git utilizado.

Avaliação

O trabalho será avaliado sob os seguintes critérios: (i) utilização correta dos conteúdos vistos anteriormente e nas aulas presenciais da disciplina; (ii) a corretude da execução dos programas implementados, que devem apresentar saída em conformidade com a especificação e as entradas de dados fornecidas; (iii) a aplicação correta de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte, e; (iv) a utilização correta do repositório Git, no qual deverá estar registrado todo o histórico da implementação por meio de *commits*. A presença de mensagens de aviso (*warnings*) ou de erros de compilação e/ou de execução, a modularização inapropriada e a ausência de documentação são faltas que serão penalizadas. Este trabalho contabilizará nota de **até 2,0 pontos na 2ª Unidade** da disciplina.

Questão 01

Crie uma hierarquia de classes de domínio para uma loja que vende os seguintes Produtos: Roupa, Bebida e Fruta. Sobrescreva o operador de inserção em stream "<<" para cada classe, de modo que o uso do mesmo imprima:

- Para Roupa: código de barras, descrição, preço, marca, sexo (M/F) e tamanho;
- Para Bebida: código de barras, descrição, preço e teor alcoólico (em %);
- Para Fruta: código de barras, descrição, preço, data do lote e validade.

Sobrescreva os operadores de adição "+" e subtração "-" de modo a permitir somar e subtrair o preço de dois produtos. Sobrescreva também o operador de igualdade "==" retornando ***true*** se dois produtos possuem o mesmo código de barras ou ***false*** em caso contrário.

Evite repetição de código. Explore, ao máximo, os conceitos de Programação Orientada a Objetos (POO) em C++ discutidos em sala de aula: classes, métodos, métodos de acesso (getters/setters), modificadores de acesso, sobrecarga de operadores, herança, polimorfismo, método virtual, classe abstrata, templates, entre outros.

Realize testes sobre as classes implementadas para mostrar o correto funcionamento de sua implementação. A carga dos produtos na lista de produtos deve ser realizada a partir dos arquivos ***.dat*** correspondentes.

SUPRESA!!!

Ainda envergonhado pela situação em que te colocou na última avaliação, Teobaldo resolveu voltar para te ajudar. Para isso, dentro de seu tempo disponível, ele te enviou um código inicial, disponível através do endereço ***https://github.com/imdcode/teobaldo.git*** e descreveu o que deve ser feito no arquivo ***README.md***.

Questão 02

Implemente em C++ as respectivas classes, atributos e métodos (incluindo construtores e destrutor) necessários para atender às seguintes abstrações:

a) Uma conta corrente possui uma agência, um número, um saldo, um status que informa se ela é especial ou não, um limite (e o limite disponível) e um conjunto de movimentações (normalmente em grande número e variável entre contas).

b) Uma movimentação possui uma descrição, um valor e uma indicação se ela é uma movimentação de crédito ou débito.

Usando as classes implementadas, escreva um programa em C++ para simular uma agência bancária. Uma agência bancária deve armazenar um conjunto de contas e permitir as seguintes operações básicas: criações de conta, exclusão de contas, saques (respeitando o saldo e o limite), depósitos, emissão de saldo e extrato, além de transferência entre contas.

Especificamente com relação à emissão de extrato, seu programa deverá exibir a lista de movimentações realizadas por meio da sobrecarga do operador de inserção em *stream* (<<).

Questão 03

Utilizando de abstração e do conceito de herança, altere o programa da questão 2 para permitir a manipulação de dois tipos de contas: conta corrente e conta poupança.