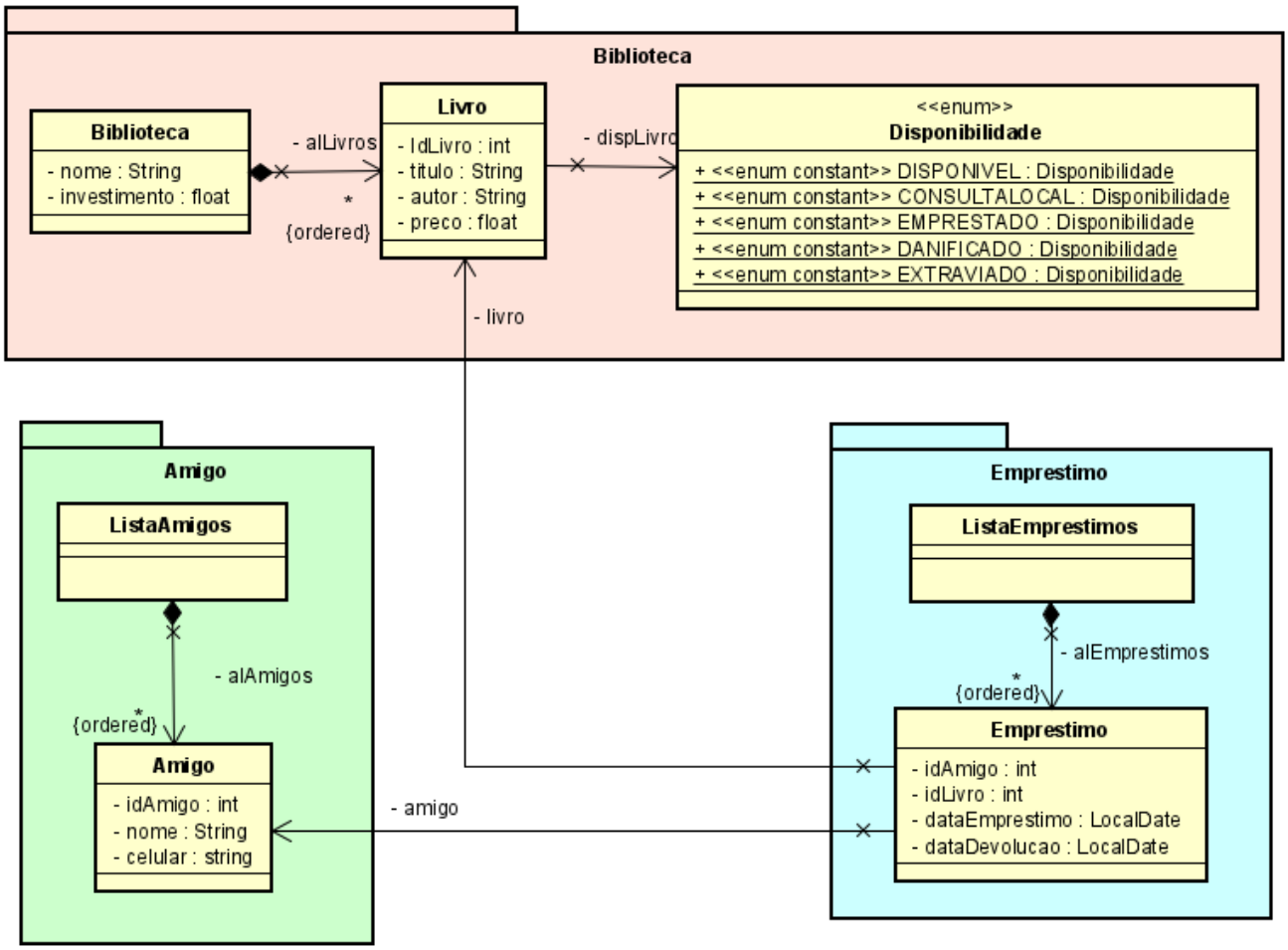
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA

CURSO DE ENGENHARIA DE COMPUTAÇÃO
PROGRAMAÇÃO ORIENTADA A OBJETOS

Sou eu próprio uma questão colocada ao mundo e devo fornecer minha resposta,
caso contrário, estarei reduzido à resposta que o mundo me der.
Carl Gustav Jung

Trabalho 1: Controle de biblioteca pessoal



Contexto

Um estudante da disciplina de POO, percebendo que sua biblioteca pessoal, cujos livros costuma emprestar aos seus amigos, estava ficando fora de controle (quem pegou? quando? já devolveu?), resolveu implementar em Java um pequeno sistema de gestão para suas necessidades de acompanhamento das movimentações de livros: cadastro de livros novos, empréstimos e devoluções etc.

O sistema proposto terá algumas classes devidamente organizadas e relacionadas permitindo operações mínimas de controle dos livros pertencentes a sua biblioteca pessoal. O projeto (modelo de classes inicial acima), cujo esqueleto foi apresentado em sala está disponível no arquivo Biblioteca.zip, já contém a base das classes com a definição dos pacotes, atributos iniciais e dos ArrayList que serão usados para manter os dados em memória. **Livro** é uma classe concreta do itens que formam a biblioteca e que podem ser empretados, estão referenciados na classe **Biblioteca** através de **alLivros**. A classe **Amigo** armazena os dados de um colega cujo conjunto está armazenado em **alAmigos** da classe **ListaAmigos**. Já a classe **Emprestimo** é responsável pela definição de um empréstimo em particular, quando um amigo (**idAmigo**) empresta um determinado livro (**idLivro**) em determinada **dataEmprestimo**. O conjunto dos empréstimos está em **alEmprestimos** da classe **ListaEmprestimos**. A classe de enumeração **Disponibilidade** empregada no atributo **dispLivro** mantém a informação do estado do livro.

Sessão Típica

Uma sessão típica consiste em:

1) cadastrar livro: efetua o cadastro de um novo livro cujo **idLivro** é gerado automaticamente (baseado na quantidade de livros de **alLivros**);

2) cadastrar amigo: efetua o cadastro de um novo amigo cujo **idAmigo** é gerado automaticamente (baseado na quantidade de amigos de **alAmigos**), incluindo o atributo **nomeAmigo**;

3) emprestar: o sistema pergunta o código do livro a ser emprestado (atributo **idLivro**), depois seleciona o amigo (pode mostrar todos para escolher o código do amigo, ou fazer busca pelo nome para identificar e pegar código **idAmigo**). Efetua o empréstimo na data corrente com objeto da classe **Emprestimo** incluído no **alEmprestimos**. Ou seja, **idAmigo** empresta livro **idLivro** em **dataEmprestimo** colocando **dispLivro** em **Livro** para **Disponibilidade.Emprestado**. As variáveis **livro** e **amigo**, da classe **Emprestimo**, referenciam diretamente os respectivos objetos (facilitando a implementação do programa), enquanto as variáveis **idLivro** e **idAmigo** são os códigos que facilitam a identificação dos objetos reais cadastrados (e servirão de chave primária em um futuro banco de dados);

4) devolver: o sistema pergunta o código do livro que estava emprestado (atributo **idLivro**), e efetua a devolução deste Item que estava fora. Ao devolver um livro, não remover o objeto de **alEmprestimos** (que portanto conterá empréstimos já concluídos – um histórico, e os empréstimos em andamento), apenas alterar a disponibilidade do livro (**dispLivro**) e atualizar o atributo **dataDevolucao**;

5) listar empréstimos atuais: o sistema mostra todos os livros emprestados: seu título, quem pegou e em que data pegou emprestado (para isso percorre o **alEmprestados** identificando quem está fora, ou seja, **dispLivro** igual **Disponibilidade.Emprestado**);

6) listar histórico de empréstimos: o sistema pergunta o código de um livro (**idLivro**) e lista todas as movimentações daquele livro, ou seja, todos os empréstimos já feitos (quem, data empréstimo e data devolução) e o empréstimo atual (se existir) para aquele livro em particular;

7) listar biblioteca: o sistema mostra todos os items ordenados pelo seu título em ordem alfabética ascendente, mostrando o título do item e sua situação (**dispItem**), preço pago pelo livro e total investido na biblioteca;

8) alterar estado: o sistema pergunta o código de um livro (**idLivro**) e permite mudar sua disponibilidade para Consulta Local (se não estiver Emprestado), ou Danificado ou Extraviado (independente de estar anteriormente Disponível ou Emprestado).

Requisitos

A organização do menu, linear como sugerido acima, ou em níveis, bem como a sequência/nomenclatura ficam livres para o estudnate decidir, o importante é permitir que sejam possíveis as operações listadas, ou seja, as funcionalidades do sistema descritas na sessão típica acima.

Para facilitar a apresentação prática do código funcionando na gravação do vídeo de entrega, deixar alguns objetos cadastrados prontos nas listas, com criação dos objetos *harcoded*, logo no início da *main*.

A tabela a seguir apresenta as características que serão consideradas (com autoavaliuação em formulário posterior):

Uso adequado das funções *static* na main com reuso quando módulos compatíveis	Funcionalidade de empréstimo de livros baseado no código do livro
Encapsulamento e métodos das classes empregados adequadamente	Funcionalidades de devolução de livros baseado no código do livro
Funcionalidades de cadastro de livro e cadastro de amigos	Alteração de Estado conforme regras especificadas e uso do enum
Funcionalidades de listagem de empréstimos, do histórico de um livro e do total de livros	Bônus: empregar tags JavaDoc adquedamente para documentação das classes de domínio da aplicação