MC322 C – Programação Orientada a Objetos
Laboratório 09 – 1s2020

Leonardo Montecchi (Professor)
leonardo@ic.unicamp.br

Iury Cleveston (PED)
iurycl@gmail.com

Matheus Mazon (PAD)
matheusmazon@outlook.com

Matheus Rotta (PAD)
matheusrotta7@gmail.com

19/05/2020

1 Exercício

O objetivo deste laboratório é explorar os conceitos de composição e delegação, e entender como composição pode ser uma forma de reutilizar código. Aproveitaremos deste laboratório também para revisar conceitos de estrutura de dados.

Observação (1). Neste laboratório você **não** deve utilizar as classes prontas do *framework Collections* do Java (ex., ArrayList). A ideia deste laboratório é de aprender os conceitos por trás das bibliotecas e não de saber utilizá-las.

Observação (2). Neste laboratório não estamos preocupados com eficiência da implementação, mas sim com organização e reuso do código. Lembre-se que, se o código for *encapsulado* corretamente, a implementação interna pode ser trocada a qualquer momento.

1.1 Lista

Implemente uma lista para guardar objetos de tipo `Pessoa` (tendo os atributos *nome*, *idade*, *cpf*). A lista deve realizar as seguintes operações:

- `adicionarInicio`, adiciona um elemento como primeiro elemento da lista.

- `adicionarFim`, adiciona um elemento como último elemento da lista.
- `adicionar`, adiciona um elemento em uma determinada posição da lista.
- `removerInicio`, remove um elemento como primeiro elemento da lista.
- `removerFim`, remove um elemento como último elemento da lista.
- `remover`, remove um elemento em uma determinada posição da lista.
- `tamanho`, retorna o número de elementos.
- `limpa`, elimina todos os elementos da lista.
- `imprimir`, imprime todos os elementos da lista.

Observação (3). Você pode implementar as estruturas da forma que achar mais adequado, mas lembre-se que as listas **não** devem ser implementadas considerando um limite de tamanho. Caso sejam implementadas com *array*, isso significa que uma vez atingido o tamanho máximo, elas devem se estender de forma transparente para o usuário.

Uma forma conveniente de implementar é como lista encadeada (*linked list*) ou lista duplamente encadeada (*doubly linked list*). Uma lista encadeada contém um determinado número de nós, cada um tendo uma referência para o elemento a ser guardado e uma referência para o próximo nó. Em uma lista duplamente encadeada cada nó possui também uma referência para o nodo anterior.

1.2 Fila

Uma fila (*queue*) é uma estrutura de dados que permite remover os elementos apenas na ordem em que foram adicionados. Ou seja, adiciona-se itens no fim e remove-se do início. Uma fila pode ser implementada de forma parecida com a implementação de uma lista. Porém, uma fila possui apenas as seguintes operações:

- `adicionar`, adiciona um elemento à fila.
- `remover`, remove o primeiro elemento da fila.
- `tamanho`, retorna o número de elementos.
- `limpa`, elimina todos os elementos da fila.
- `imprimir`, imprime todos os elementos da fila.

Implemente uma fila para guardar objetos de tipo `Pessoa`, reaproveitando o código da lista.

1.3 Fila de Atendimento

Implemente uma fila de atendimento para `Pessoas`. A fila deve suportar duas prioridades de atendimento: *normal* e *prioritário*. Pessoas com atendimento prioritário são removidas primeiro, em ordem de chegada; pessoas com atendimento normal são removidas também em ordem de chegada, mas apenas quando não tiver nenhuma pessoa com atendimento prioritário.

Esse problema pode ser visto como um caso particular de fila de prioridade (*priority queue*), em que apenas dois valores de prioridades são permitidos. A fila de atendimento possui as seguintes operações:

- `adicionar`, adiciona uma pessoa com atendimento *normal* à fila.
- `adicionarPrioritario`, adiciona uma pessoa com atendimento *prioritário* à fila.
- `remover`, remove a primeira pessoa da fila.
- `tamanho`, retorna o número de pessoas na fila.
- `imprimir`, imprime todas as pessoas na fila, distinguindo se elas estão com atendimento prioritário ou normal.

1.3.1 Reuso por Composição

Implemente a fila de atendimento reutilizando o código da fila de `Pessoas` por meio da composição. *Sugestão:* Utilize duas filas diferentes.

1.3.2 Reuso por Herança

Uma fila de atendimento *é uma* fila. Implemente a fila de atendimento reutilizando o código da fila de `Pessoas` por meio da herança.

1.4 Diagrama UML

Desenhe um diagrama UML do código implementado (ambas as versões). Algumas ferramentas para desenhar diagramas UML:

- Eclipse Papyrus: `https://www.eclipse.org/papyrus/`
- Umbrello: `https://umbrello.kde.org/`
- StarUML `http://staruml.io/`

2 Submissão

Submeta o trabalho no Google Classroom da disciplina, em formato de arquivo compactado (.zip). O **nome do arquivo** e do **projeto Java** devem estar no formato **raLab09**.

- O que submeter: Um arquivo compactado contendo o código fonte e o(s) diagrama(s) UML criados.
- Este laboratório **não vale nota**.