Universidade Federal de Campina Grande – UFCG

Centro de Engenharia Elétrica e Informática – CEEI

Departamento de Sistemas e Computação – DSC

Disciplina: Laboratório de Programação 2
Período: 2011.1

Professores: Livia Campos (turma1)
Reinaldo Gomes (turma2)
Nazareno Andrade (turma 3)

Laboratório 06

Neste laboratório iremos praticar o uso de coleções do tipo lista. Além disso, estimular a realização de testes de unidade e documentação e continuar desenvolvendo habilidades na definição da lógica de controle de programas Java.

Instruções:

- Data de entrega
  - Parte 1: 24/03/2011 até 12h (Turmas 1 e 2) ou 25/03/2011 até 10h (Turma 3);
  - Parte 2: 31/03/2011 até 12h (Turmas 1 e 2) ou 01/04/2011 até 10h (Turma 3);
- Crie o projeto lab06 no eclipse e programe todos os exercícios dentro do pacote lp2.lab06;
- **Salve o projeto** em um arquivo zip chamado lab06-<seuNome>.zip (ou .tgz ou .rar) e o envie para juiz.lp2@gmail.com Use como *subject* do email lab06-<seuNome> (turma 1, 2 ou 3). **A partir desse lab, nomeações incorretas dos arquivos (e subject do email) a serem enviados serão penalizadas na nota do lab**;
- Certifique-se de que seus programas não têm erros de compilação;
- Antes da definição de cada classe escreva o seu nome em comentário (/*Aluno: <seu nome>*/);
- Não copie o programa do seu vizinho. Se tiver dúvida converse com o professor ou com um monitor.

Parte 1: Implementando uma coleção com arrays

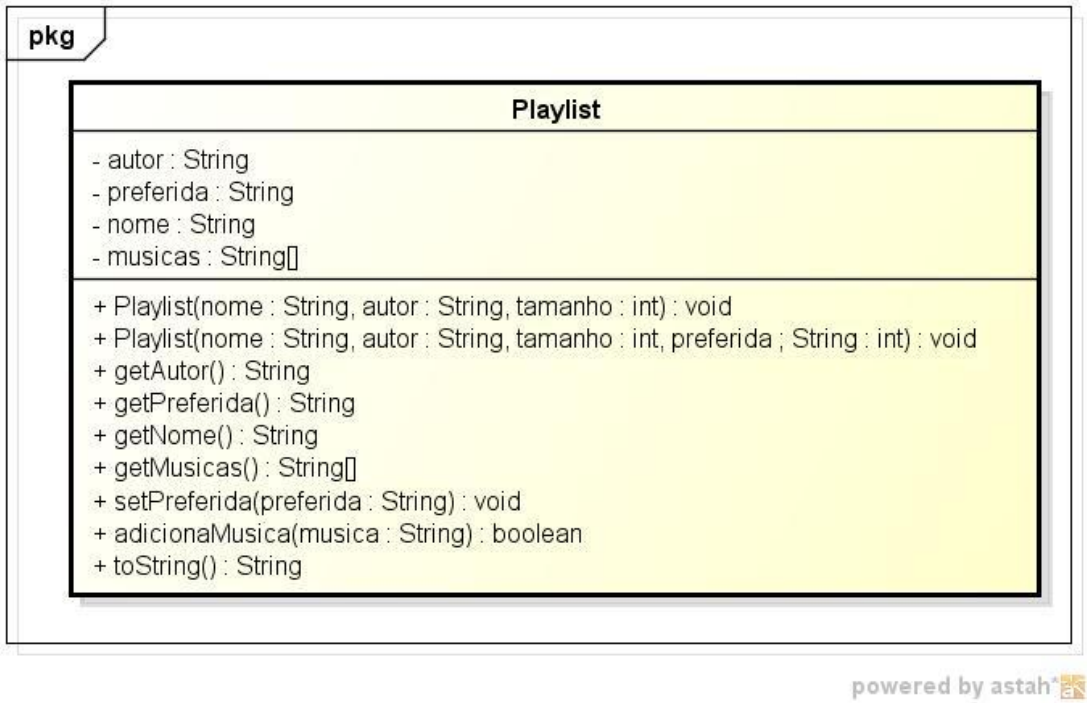
1. Escreva a classe Playlist que representa uma playlist feita por alguém, com quantidade limitada de músicas. Cada playlist possui como atributos o autor da playlist, nome da playlist, uma música preferida, uma coleção de M músicas (array de tamanho M, onde as músicas são representadas pelos seus nomes). Nome e autor não podem ser vazios. O valor de M (M > 0) é definido no momento da criação de um objeto do tipo PlayList. Sua classe pode ter outros atributos se assim for necessário. Escreva a classe especificada, incluindo (ver também o UML abaixo descrevendo estas funcionalidades):

(i) Dois construtores: o primeiro recebe nome, autor da playlist e M, enquanto o segundo recebe nome, autor da playlist, M e música preferida;

(ii) Métodos acessadores para os atributos da classe, e um método modificador para a música preferida. Este método modificador adiciona a nova música preferida na coleção de músicas, caso a mesma ainda não esteja lá (observar limitação de tamanho da coleção); se a coleção já estiver cheia deve-se lançar uma exceção;

(iii) Método que adiciona músicas à playlist. O método de adicionar músicas recebe como parâmetro o nome da música que deve ser adicionada no conjunto de M músicas na próxima posição livre (ex. na primeira vez que o método for chamado, a música será adicionada na posição 0 da coleção; na segunda vez, na posição 1 e assim por diante). Se o usuário tentar cadastrar mais músicas do que a quantidade máxima de músicas permitida (M), o método deve retornar false, sem adicionar a nova música; caso contrário, a música será adicionada e o método deve retornar true;

(iv) Não esqueça do método toString().



Parte 2: Implementando uma coleção com ArrayList<E>

1. Crie a classe PlayListDinamica que representa uma playlist sem limitações de tamanho. Esta classe possui todas as funcionalidades da classe PlayList (desenvolvida na primeira parte deste lab), no entanto, usa uma List<String> (ArrayList<String>) para armazenar sua coleção de músicas.
2. Modifique a classe PlayListDinamica, adicionando as seguintes funcionalidades (ver também o UML abaixo descrevendo estas funcionalidades):
   (i) Método que adiciona uma música na i-ésima posição da playlist. O método deve lançar uma exceção caso o valor de i não seja válido;
   (ii) Método que pesquisa uma música na playlist a partir de uma chave (nome da música). Quando a música pesquisada não existir retorna null, caso contrário, retorna a música encontrada;

(iii) Método que pesquisa se uma determinada música pertence à playlist. Nesse caso, a chave de pesquisa é o nome da música. Quando a música pesquisada não existir retorna false;
(iv) Método que retorna a i-ésima música da playlist, onde i é passado como parâmetro. Se o valor de i não for válido retorna null;
(v) Método que remove uma música da playlist a partir de uma chave (nome da música). Quando a música não existir retorna null, caso contrário, retorna a música removida;
(vi) Método que retorna a quantidade de músicas atualmente na playlist;
(vii) Método equals(...) para comparar duas playlists. Duas playlists são iguais se tiverem o mesmo conjunto de músicas, incluindo número de músicas repetidas. Uma playlist que tem as músicas [A, B, B, C] é diferente da playlist [A, B, C]. As playlists [A, C, D, B] e [B, A, C, D] são iguais.

Observações para a segunda parte do lab:

1. Crie testes de unidade para a classe PlayListDinamica.
2. Inclua comentário javadoc na classe implementada.