Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital
IMD0040 - Linguagem de Programação 2
Aula05 - Lista de exercícios

- Esta lista de exercício é composta por 4 questões. Sendo uma fácil, uma média, uma difícil e uma de desafio.
- O aluno deverá apresentar ao professor até o término da aula, e submeter via SIGAA, a solução de uma das questões de dificuldade média.

Questão Fácil – Sistema de Correio.

A tarefa

Abra o projeto mail-system (do capítulo 3). A ideia deste projeto é simular o ato de enviar itens de e-mail entre usuários. Um usuário utiliza um cliente de correio para enviar itens de e-mail a um servidor a fim de que elas possam ser entregues a outro cliente de correio.

Primeiro crie um objeto do tipo MailServer. Agora crie o objeto MailClient para um dos usuários. Ao criar o cliente, você precisará fornecer a instância MailServer como um parâmetro. Utilize aquela que você acabou de criar. Você também precisa especificar um nome de usuário para o cliente de correio. Agora crie um segundo MailClient de maneira similar mas com diferente nome de usuário.

Experimente

Crie objetos do tipo MailClient. Eles podem enviar itens de correio de um cliente de correio para outro (utilizando o método sendMailitem) e receber mensagens (utilizando o método getNextMailItem).

Diagrama

Desenhe um diagrama de objetos (no caderno) da situação que você tem depois de criar um servidor de correio e três clientes de correio.

Melhorando o sistema

Adicione uma linha de assunto a um correio eletrônico. Certifique-se de que, ao imprimir mensagens, a linha de assunto também seja impressa. Modifique o cliente de e-mail de maneira correspondente.

Questão Média – Contagem de palavras

A tarefa

Você faz parte de uma equipe que está desenvolvendo um sistema para uma editora de livros. Como parte desse sistema, é necessário desenvolver uma classe que seja capaz de contar as palavras de um determinado texto.

Com base nos seus conhecimentos de Strings, implemente um método que faça a contagem de cada palavra em um dado texto (com exceção dos espaços) recebido como uma String, e as imprima de acordo com o padrão abaixo:

<**Palavra>: <quantidade>**

Por exemplo, dado o texto abaixo

"eu tenho um barco vermelho e um carro vermelho"

, o seu programa deve imprimir:

```
eu: 1
tenho: 1
um: 2
barco: 1
vermelho: 2
e: 1
carro: 1
```

Dica: Verifique a API da classe String. Considere também o funcionamento do projeto TechSupport apresentado no capítulo 5 (ou capítulo 6, dependendo da versão do livro).

Questão Médio – Manipulação de um Array de Strings.

A tarefa

Imagine que você tem uma vasta coleção de arquivos de som que você deseja manter. Sua tarefa é escrever uma classe que seja capaz de armazenar uma lista de nomes de arquivos (seus arquivos de som) e que permite a você manter essa lista: adicionar/remover nomes de arquivos da lista, listar o conteúdo de sua lista, etc. Infelizmente, você não será capaz de salvar sua lista entre diversas execuções de sua classe, mas isso pode ser facilmente adicionado no futuro (não sendo necessário para este trabalho).

Para facilitar seu trabalho, você terá acesso a uma classe chamada SoundEngine, que é capaz de executar um arquivo de som dado o nome do arquivo como uma String. Essa classe pode tocar arquivos no formato MP3). Sua classe deverá ser capaz de utilizar a classe SoundEngine para tocar um arquivo selecionado da lista, ou a lista completa. A pasta de projeto para este trabalho é chamada MusicPlayer, e está disponível no sigaa. Esse projeto contém uma subpasta chamada áudio com alguns arquivos de áudio para seus testes.

Métodos

Escreva uma classe que utilize um ArrayList para armazenar uma coleção de objetos do tipo String. Não deve existir um limite para o número de Strings que podem ser armazenados. Cada objeto String representa o caminho para um arquivo de áudio. Os caminhos para os arquivos podem ser armazenados de forma relativa, representando arquivos a partir da pasta atual do sistema (“audio/BoB-Airplane.mp3”) ou de forma absoluta, permitindo identificar o arquivo a partir da raiz do sistema de arquivo de seu computador (“/home/usuário/downloads/audio/Beck-HitInTheUsa.mp3”). Sua classe deve oferecer os seguintes métodos:

void addFile(String file){ // um método que recebe uma String e a armazena na coleção.

void getNumberOfFiles(){ // um método que retorna o número de Strings na coleção.

void listFiles(){ /* um método que imprime as Strings que estão na coleção, uma por linha, e com um número de índice antes de cada nome de arquivo; por exemplo:
0: “/audio/BoB-Airplane.mp3”.
1: “/home/usuário/downloads/audio/Beck-HitInTheUsa.mp3”. */

void removeFile(String file){ /* um método que remove o item que se encontra naquela posição da coleção. Se o índice não for válido, então uma mensagem de erro deve ser impressa e a coleção não deve ser alterada. */

void playFile(String file){ /* um método que usa um objeto SoundEngine para tocar o arquivo de nome passado por parâmetro. Use o método playCompletely do objeto SoundEngine. Por exemplo, se a variável engine referencia um objeto SoundEngine, então: **engine.playCompletely("audio/Gunther-DingDingDongSong.mp3");** */

void playAll(){ /* um método que faz com que todos os arquivos da coleção sejam tocados, um após o outro, usando um objeto SoundEngine. Utilize o método playCompletely para tocar os arquivos. */

void removeFile(String file){ /* um método que remove da coleção o primeiro arquivo que contêm o parâmetro recebido como uma substring. Por exemplo, chamando o método removeFile("Beck") poderá remover o arquivo "audio/Beck-HitInTheUsa.mp3". Se mais de um arquivo da coleção contiver a String recebida como parâmetro, então somente o primeiro deverá ser removido. Imprima uma mensagem de erro no caso em que nenhum arquivo da coleção tenha a String passada como parâmetro. Perceba que deverão existir dois métodos chamados removeFile na sua classe. Você poderá distinguir entre eles através dos tipos de parâmetro que eles recebem. */

Finalizando a classe

Não se esqueça de ter um método construtor! Ele é extremamente necessário e deve preencher o campo quando o objeto for criado (lembre-se: o campo deve ser declarado como um ArrayList<String> e, apenas no construtor, deve ser estanciado corretamente).

*Dica: A API é sua amiga, utilize-a caso seja necessário. Procure por java.util.ArrayList.

Questão Difícil – Subset Sum

A tarefa

Subset sum é um algoritmo de soma em arrays. Ele varre o array procurando um pequeno subarray cuja soma seja um elemento qualquer definido pelo usuário. Exemplo: Quero encontrar o subarray neste array que some 9.

lista = {4, 2, 3, 5, 1, 0}.

Logo, o subarray que forma o 9 é formado por 3, 5, 1.

Execução

Esta tarefa não necessita de muitos métodos, apenas um campo ArrayList que deve ser iniciado em seu construtor e um método para fazer o cálculo.

returnType subsetSum(int x){

Escolha o tipo de retorno, ela pode retornar outro ArrayList, não retornar nada e apenas imprimir na tela ou retornar um vetor de inteiros. Por exemplo, no teste acima, o usuário pode retornar um ArrayList<int> saida, que armazena os valores 3, 5 e 1. Pode retornar um vetor de inteiros com 3, 5 e 1. Pode retornar um vetor com os indíceis do ArrayList (2, 3 e 4). O usuário pode até mesmo apenas imprimir na tela os valores (sim, utilizando System.out.println).

Como o retorno vai acontecer, não importa. Apenas deixe explícito do que se trata quando as informações forem exibidas.

Para se pensar (optativo): Utilizando seus conhecimentos de EDB2, você consegue dizer qual a complexidade do algoritmo desenvolvido?