Prática 12 Maven e Revisões

Tópicos

- Introdução ao Maven
- Distribuição de aplicações Java
- Revisões

Objetivos

- Gerir dependências no Java usando o Maven, seus repositórios, e plugins
- Preparar a distribuição de uma aplicação num ficheiro JAR autocontido
- Usar expressões de transformação de dados ou explorar processamento em Stream sempre que adequado
- Revisões para o mini-teste prático *(semana de 26/05 a 29/05)*

Exercício 12.1 Maven *(continuação TP)*

Instale o maven a partir do seu gestor de pacotes (Linux), sistema de gestão de pacotes do tipo Chocolatey (Windows) ou Homebrew (MacOS), ou descarregue os binários da página oficial e coloque-os numa pasta acessível ao PATH configurado no Visual Studio Code.

Siga o tutorial nos slides da aula teórico prática correspondente para:

1) Criar ficheiros JAR para a distribuição da vossa aplicação a utilizadores finais.
2) Criar diferentes tipos de JARs, ficheiros mais pequenos se as dependências forem resolvidas externamente, ou ficheiros mairores ("fat JARS") se contiverem todas as dependências.
3) Utilizar Maven para resolver dependências de uma aplicação Java, compreendendo qual o seu lifecycle, e funcionalidades relevantes.

Exercício 12.2 Lista de contatos *(revisões)*

Escreva um programa que permita gerir uma lista de contatos. A classe "Contact" permite representar cada contato que irá fazer parte da lista. O programa deve usar Java Collections para gerir os contactos.

Cada contato deve conter os seguintes atributos:

- Identificador único (int) – deve ser automático, incremental
- Nome da pessoa (String)
- Número de Telemóvel (int)
- E-mail (String)
- Data de Nascimento (LocalDate)

O programa deve conter, pelo menos, as seguintes classes:

- Contact é a classe que representa cada contacto na lista de contatos. Esta classe deve ter os atributos mencionados acima, um construtor, função toString e os setters e getters apropriados.
- ContactManager é a classe responsável por gerir as pessoas na lista de contactos. Deve implementar os seguintes métodos:
    - addContact(Contact c): Adiciona um novo contato ao sistema. Não permita adicionar o mesmo contato mais do que uma vez.
    - removeContact(int id): Remove um contato do sistema usando o seu identificador único.
    - getContact(int id): Obtém um contacto no sistema com base no identificador. Retorna o pedido ou null se o contato não existir.
    - calculateContactCost(int id): Calcula e retorna o custo de contato usando a StandardCostCalculator. No caso de não existir o contato, retorna -1.
    - printAllContacts(): Imprime a informação de todos os contatos que estão no sistema.
    - readFile(String file): Importa a informação de um conjunto de contatos a partir de um ficheiro. Estes contatos devem ser adicionados aos já existentes.
    - writeFile(String file): Escreve a lista de contatos existentes no ContactManager para um ficheiro. O ficheiro deve conter todos os dados armazenados no ContactManager separados por ";".
- StandardCostCalculator é a classe responsável por calcular o custo de contactar determinada pessoa, de acordo com o tipo de contacto, e implementa a interface IContactCostCalculator. O tarifário será o seguinte:
    - 0 €/unidade se for email
    - 0.10 €/unidade se for um número de telemóvel

Utilize o ficheiro ContactTester.java para testar e demonstrar o uso da classe ContactManager *(não precisa de menu).* Pode editar este ficheiro livremente, mas deverá assegurar-se que este continua a testar as seguintes funcionalidades:

- Cria uma nova instância de ContactManager.
- Adiciona contatos ao sistema.
- Editar contatos no sistema
- Remove contatos do sistema.
- Lê a partir de um ficheiro os dados referentes a vários contatos.
    - O programa deve ser capaz de ler o ficheiro contatos.txt
- Obtém um contato em específico.
- Lista todos os contatos no sistema.
- Importar contactos de um ficheiro:
    - Ao executar o programa, deve ler o ficheiro "contactos.txt" e iniciar a lista telefónica com os dados aí guardados.
- Exportar contatos para um ficheiro:
    - Ao terminar a execução, o programa deve escrever o conteúdo atual da lista para um ficheiro, respeitando o formato original.

Extras:

- Não deve ser possível adicionar um contacto que já exista. Para verificar se dois contactos são iguais deve considerar o nome, o telemóvel e e-mail.
- Deverá validar o e-mail (usando uma expressão regular simples) e o contacto telefónico (deve ter 9 dígitos).
- ***Considere uma funcionalidade de pesquisa na estrutura de onde, pex., devolve todos os contatos que nasceram depois de 01/01/2000. Implemente essa funcionalidade usando os princípios de Stream Processing e expressões lambda.***

Informação adicional

```
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
LocalDate date = LocalDate.parse("2023-05-01", formatter);
System.out.println(date.format(formatter));
```