

Fundamentos de Java

Exercícios Propostos

Expressões Regulares

www.softblue.com.br

# Exercício

Crie uma aplicação para organizar arquivos de fotos. Suponha que você tem algumas fotos tiradas com a sua câmera digital armazenadas no seu computador e que os nomes dos arquivos são os seguintes:

IMG00025-20111002\_Florianópolis.jpg

IMG00026-20111003\_São Paulo.jpg

IMG00027-20111110.jpg

IMG00028-20111110.jpg

Se você olhar atentamente para o nome do arquivo, verá que ele traz a string “*IMG*” seguida de um número sequencial (ID da foto). Depois do hífen, o arquivo traz a data da foto, no formato “*AAAAMMDD*”. Por fim, caso a câmera tenha identificado uma localidade (nem todas as fotos possuem esta informação), ela é armazenada após o caractere “\_”. A extensão do arquivo é sempre “jpg”.

A aplicação deve ter algumas classes, que devem ser definidas seguindo o modelo abaixo:



A classe Foto representa uma fotografia tirada pela câmera. Possui um ID, uma data e um local (se a câmera não registrou o local, ele deve ser null). Ela possui também os métodos *getters* e *setters* correspondentes. A classe Album agrupa uma ou mais fotos. O método adicionar() insere uma nova foto na lista de fotos do álbum, enquanto o método listarFotos() imprime os dados das fotos do álbum no console. A classe Organizador é quem possui a lógica de criar o álbum de fotos através da leitura dos arquivos. Esta lógica é implementada pelo método carregarFotos(), que recebe como parâmetro o diretório onde estão as fotos a serem adicionadas no álbum e retorna o álbum criado.

Para testar a aplicação, crie uma classe com um método main() que instancia um Organizador, cria um álbum com base em algumas fotos e depois imprime os dados das fotos do álbum (você pode criar arquivos fictícios de fotos seguindo o padrão da nomenclatura mostrado anteriormente para poder testar).

**Dica**: utilize a classe Scanner para executar a divisão do nome do arquivo em *tokens*.

# Exercício

Crie uma classe Validador com os métodos validarCPF() e validarTelefone(), que validam um CPFs e telefones, respectivamente. Os métodos devem receber como parâmetro a String a ser validada e retornar true se a validação foi feita com sucesso ou false, caso contrário.

O validador de CPF deve aceitar o padrão que usa os pontos separadores e o hífen, que separa os dígitos verificadores. Outros padrões aceitos são: sem nenhum separador e também com espaços em branco no lugar dos pontos e do hífen. Veja alguns exemplos de CPFs que devem ser considerados válidos:

* **432.567.324-01**
* **43256732401**
* **432 567 324 01**

O validador de telefone deve aceitar telefones de oito dígitos com hífen ou espaço em branco separando os quatro primeiros dos quatro últimos dígitos. Telefones com o DDD colocado entre parênteses também devem ser suportados. Veja alguns exemplos de telefones que devem ser considerados válidos:

* **5634-2395**
* **5634 2395**
* **(99) 5634-2395**

**Dica**: Você pode utilizar o método matches() da classe Matcher, que retorna true se o texto fornecido é compatível com a expressão regular ou false, caso contrário.