3ª AVALIAÇÃO

Data	Sexta-feira, 7 de julho de 2023	Duração	2:45	Valor	10,0

☞ **ATENÇÃO:** 1. Leia, cuidadosamente, toda a prova antes de começar.
2. A interpretação das questões é parte integrante da prova.
3. Qualquer tentativa de fraude - "cola" - será punida com nota zero.
4. O valor de cada questão está entre parênteses.

- Critérios de Avaliação

1. Verifique se a versão da JRE instalada na IDE Eclipse e a versão do compilador Java corresponde ao JDK 19.

2. Crie um único projeto Java na IDE Eclipse para resolver todas as questões desta prova. O projeto deve ter o seu nome e sobrenome sem espaços, por exemplo: NelsonMandela

3. ~~Crie uma pastar jar dentro do diretório do projeto e copie todos os arquivos JAR fornecidos como anexo para essa pasta. Em seguida, atualize o projeto no Eclipse e adicione cada arquivo JAR ao projeto (*Build Path*).~~

4. Ao terminar a prova compacte o projeto para criar um arquivo 7z com o seu nome e sobrenome, por exemplo: NelsonMandela.7z

5. Desenvolver as classes usando os conceitos fundamentais de orientação a objetos.

6. Incluir em cada classe apenas as definições de tipos de dados, variáveis, constantes e métodos que forem essenciais para a funcionalidade da mesma.

7. Escrever o código fonte obedecendo o princípio do menor privilégio.

8. Escrever métodos específicos, ou seja, com papel (função) claro e objetivo. Lembre-se das orientações apresentadas durante as aulas, ou seja, faça as quatro perguntas para auxiliar na criação dos métodos.

9. Não escrever código redundante.

10. É proibido modificar os identificadores, as definições e/ou implementações de classes, interfaces e métodos fornecidos no texto ou como anexo.

11. É permitido acrescentar novas declarações e/ou implementações de classes, interfaces, métodos, variáveis e constantes desde que estejam de acordo com os critérios acima.

12. Criação de identificadores significativos para aprimorar a inteligibilidade do código fonte.

13. Criação de pacotes para organização dos tipos de dados do projeto. Não é permitido usar os mesmos nomes de pacotes dos projetos criados durante a aula. Use nomes significativos.

- Questão

1. (1,0) Implemente os métodos abaixo em uma interface chamada Pesquisa.

```
/*
    Pesquisa um elemento na lista.
*/
public default <T extends Comparable<T>> Optional<T> pesquisar(List<T> lista, T elemento);

/* Pesquisa um elemento no vetor. Se o elemento for encontrado retorna a sua posição no vetor, se não
   retorna -1. O valor de retorno tem que ser encapsulado em um objeto OptionalInt.
 */
public default <T extends Comparable<T>> OptionalInt pesquisar(T[] vetor, T elemento);
```

2. (3,0) Crie um programa para testar os métodos da interface Pesquisa criada na Questão 1. Deve ser criado dois testes para cada método de pesquisa, cada um com uma lista ou vetor de objetos de tipo diferente, e todas as chamadas do método pesquisar deverão ser realizadas de modo polimórfico. Deve-se criar também um método genérico chamado testarPesquisa para:

1. receber a estrutura de dados (vetor ou lista) e o elemento a ser pesquisado;
2. executar o teste de pesquisa;
3. retornar o valor com o resultado da pesquisa.

O método genérico testarPesquisa deve ser capaz de fazer testes de pesquisa sobre qualquer vetor ou lista de objetos.

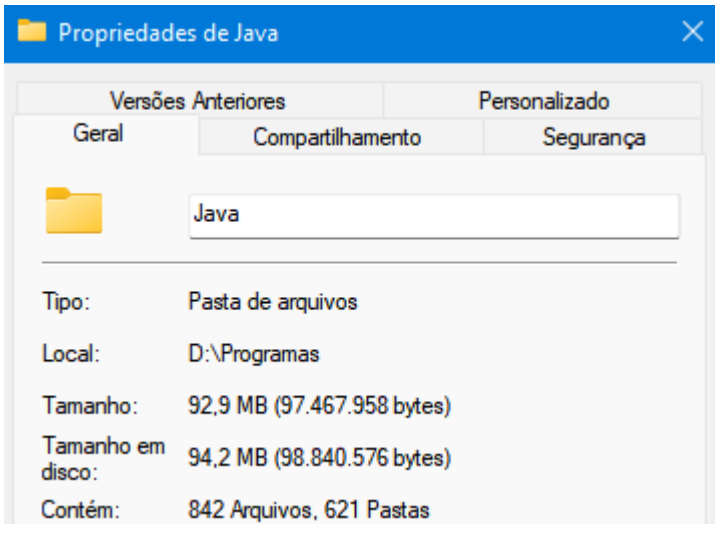
3. (6,0) Desenvolva um programa *multithreaded* que calcule o tamanho real, tamanho ocupado em disco, o número de arquivos e subdiretórios de um diretório fornecido pelo usuário. Considerar o tamanho em *bytes* de cada arquivo e subdiretório da pasta indicada, incluindo todos os subdiretórios, como se pode observar na imagem abaixo ao ler o conteúdo do campo Contém que informa o número total de arquivos e subdiretórios (pastas) do diretório Java. Use a classe java.io.File para obter os dados necessários.

Um arquivo no sistema de arquivos NTFS do Windows com tamanho real de um byte ocupa um tamanho em disco igual ao tamanho de um *cluster*, que é igual a 4.096 bytes.

Crie uma classe Tamanho para ser usada como uma área de memória compartilhada entre as *threads* e para armazenar o tamanho de cada arquivo.

Essa classe deve usar um mapa (java.util.Map) para associar o nome do arquivo (chave) com o seu tamanho (valor).

A classe Tamanho deve ser o monitor usado na sincronização entre as *threads* produtora e consumidora e ser implementado usando wait e notify.



A *thread* produtora ObterDiretorio deve obter o nome e o tamanho dos arquivos e inserir no objeto monitor Tamanho.

A *thread* consumidora CalcularTamanhoDiretorio deve recuperar do objeto monitor o tamanho de cada arquivo usando o seu nome e totalizar o tamanho real e ocupado em disco de cada subdiretório até que todos os arquivos e subdiretórios do diretório sejam processados. Essa *thread* deve calcular também o número total de arquivos e subdiretórios.

O programa deve exibir sua saída usando o leiaute abaixo:

Tamanho real: 97.467.958 bytes
Tamanho em disco: 98.840.576 bytes
Arquivos: 842
Subdiretórios: 621

Boa Prova!