Classes, Objetos e Arrays

1. Crie uma classe chamada **ArrayDeInts**. Essa classe deve ter como atributo um array do tipo **int** de tamanho **10**.

2. Crie um construtor que inicialize os valores do array encapsulado de **forma randômica** com números de **1 a 50**.

3. Crie um método chamado **imprime** que exibe todos os valores do array encapsulado.

4. Crie um método chamado **leia** que solicita e lê os valores informados pelo usuário e os salva no array encapsulado.

5. Crie um método chamado **ehIgual** que recebe um array do tipo **int** como argumento e retorne **true** se os tamanhos e valores dos arrays forem iguais. Veja abaixo como receber um array por argumento:

```
boolean ehIgual(int[] meuarray){
        …
}
```

6. Crie um método chamado **soma** que recebe um valor inteiro e some esse valor a cada elemento do array encapsulado.

7. Crie um método chamado **total** que retorne a soma de todos os elementos do array encapsulado.

8. Crie um método chamado **existe** que recebe um valor do tipo **int** como argumento e retorne **true** se o valor existe no array encapsulado.

9. Crie um método chamado **leiaUnico** que solicita e lê os valores informados pelo usuário e os salva no array encapsulado. Esse método não deve permitir inserir valores repetidos, ou seja, ele deve informar ao usuário que aquele valor já existe e solicitar que digite um outro valor.

10. Crie um método chamado **media** que retorne a média dos valores do array encapsulado, um método chamado **maior** que retorne o maior valor do array encapsulado e um método chamado **menor** que retorna o menor valor do array encapsulado.

11. Crie um método chamado **somaArray** que recebe um array do tipo **int** como argumento e acumule os valores do array passado como argumento ao array encapsulado. Essa operação somente poderá ser feita se os arrays tiverem o mesmo tamanho. Se o tamanho for diferente, o array encapsulado não deverá ser modificado.

12. Crie um método chamado **troca** que recebe dois valores inteiros como argumentos e troca os valores nas posições especificadas pelos argumentos. Por exemplo, se os valores do array encapsulado forem {3, 2, 8, 9, 1, 4} e o método troca for chamado com os argumentos 1 e 4, os elementos nesses índices serão trocados de forma que os valores do array encapsulado serão {3, 1, 8, 9, 2, 4}.

13. Crie um método chamado **maisProximo** que recebe um valor do tipo int como argumento e retorna o valor do array encapsulado que seja mais próximo (ou seja, cuja diferença seja a menor) do valor passado. Por exemplo, se o array encapsulado for {0, -2, -4, 10} e o argumento para o método for 6, o método deverá retornar 10.

14. Crie um método chamado **inverte** que inverte a ordem dos elementos do array encapsulado, de forma que o primeiro passe a ser o último e vice-versa. Por exemplo, se o array encapsulado for {9, 9, 2, 7, 0, 5}, depois da execução do método ele será {5, 0, 7, 2, 9, 9}.

15. Crie um método chamado **ehPalindromo** que retorna true se o array encapsulado for palíndromo. Um array palíndromo é aquele que pode ser lido do início para o fim e do fim para o início, da mesma forma. Por exemplo, o array (2, -4, 9, 0, 9, -4, 2) é palíndromo.

16. Crie um método chamado **ehCrescente** que verifica se os elementos de um array estão ordenados crescentemente, comparando cada elemento do array com seu próximo, retornando true se todos os elementos forem menores que os seus respectivos próximos ou false se qualquer um for maior do que o próximo.

17. Crie um método chamado **produtoEscalar** que retorne um valor do tipo int que seja o produto escalar do array encapsulado e de um outro array de inteiros passado como argumento. Por exemplo, se o array encapsulado for {9, 2, -6, 7, 0} e o passado como argumento for {1, -4, 5, 9, 2}, o produto escalar será $9\times1+(-2\times4)+(-6\times5)+7\times9+0\times2 = 34$.

18. Crie um método chamado **existeQualquer** que recebe um outro array de inteiros como argumento e retorna true se qualquer um dos elementos do array passado como argumento existir no array encapsulado.

19. Crie um método chamado **existemTodos** que recebe um outro array de inteiros como argumento e retorna true se todo os elementos do array passado como argumento existir no array encapsulado, em qualquer ordem e independentemente de repetições.

20. Crie um método chamado **distanciaEuclidiana** que retorne um valor do tipo float que seja igual à distância euclidiana acumulada entre o array encapsulado e de um array passado como argumento. A distância euclidiana entre dois arrays numéricos é definida como sendo a raiz quadrada das somas dos quadrados das diferenças de seus elementos. Por exemplo, se o array encapsulado for {1, 3, 0} e o passado como argumento for {1, 9, -4}, a distância euclideana será igual a $\sqrt{(1-1)^2+(3-9)^2+(0-(-4))^2}=\sqrt{0+36+16}\approx 7{,}2111$.