Paradigmas da Programação
Teórico-Prática 5
Exceções e ArrayLists

Objetivos Específicos: **Lançamento**, **captura** e **tratamento** de exceções. Classes e objetos de exceção. Hierarquia de classes de exceção. Exceções **nativas** e exceções **próprias**. Exceções ***unchecked*** e exceções ***checked***. Cláusulas ***try***, ***catch*** e ***throws.*** Instrução ***throw***.

Classe ***ArrayList***. Repetição ***foreach***.

Exercício

1. Criar um novo **projeto** *Netbeans* chamado ***Pessoa***.

2. Criar a seguinte versão simplificada da classe ***Pessoa.***

```
public class Pessoa {

    private String nome;
    private int numeroBI;

    private final static String NOME_POR_OMISSAO = "sem nome";

    public Pessoa(String nome, int numeroBI) {
        this.nome = nome;
        this.numeroBI = numeroBI;
    }

    public Pessoa() { this(Pessoa.NOME_POR_OMISSAO, 0); }
    public Pessoa(Pessoa outraPessoa){ this(outraPessoa.nome, outraPessoa.numeroBI); }

    public String getNome() { return this.nome; }
    public int getNumeroBI() { return this.numeroBI; }

    public void setNome(String nome) { this.nome = nome; }
    public void setNumeroBI(int numeroBI) { this.numeroBI = numeroBI; }

    @Override
    public String toString() {return this.nome + " tem o BI número " + this.numeroBI;}

    @Override
    public boolean equals(Object obj) {
        if (this == obj) { return true; }
        if (obj == null || this.getClass() != obj.getClass()) { return false; }
        Pessoa p = (Pessoa) obj;
        return this.nome.equalsIgnoreCase(p.nome) && this.numeroBI == p.numeroBI;
    }
}
```

3. Modificar a classe ***Pessoa*** de modo a **validar os argumentos** relativos aos atributos de instância: ***nome*** e ***numeroBI***. No caso de esses argumentos serem inválidos, deve ser lançada uma exceção ***unckecked*** do tipo ***IllegalArgumentException***. Considerar que o número do BI deve ser um número inteiro com 8 dígitos no máximo.

4. Criar uma **classe principal** chamada ***TestePessoa*** para testar a classe ***Pessoa***.

5. Nesta nova classe, criar um **programa** com interface de utilizador do tipo **consola**, para **construir e visualizar** uma instância de Pessoa cujo **nome** e **número de BI** são definidos pelo utilizador. Assuma que o utilizador só introduz **dados válidos**.

6. Testar o programa implementado com **dados válidos**.

7. Testar novamente o programa com um **nome inválido**: *string* vazia.

8. Corrigir o programa para **não abortar** a sua **execução**.

9. Testar novamente o programa com um **nome válido** e um **número de BI inválido**: -1.

10. Testar novamente o programa com um **nome válido** e um **número de BI inválido**: 11a.

11. Corrigir novamente o programa**.**

12. Criar uma classe de exceção **própria *unchecked*** e derivada da classe ***IllegalArgumentException***. Esta nova classe deve chamar-se ***ArgumentoForaDosLimitesException*** e servirá para personalizar a exceção gerada por argumentos inválidos do número do BI.

13. Copiar a classe ***Pessoa*** e dar o nome ***Pessoa2***.

14. Modificar a classe ***Pessoa2*** de modo a usar a nova classe de exceção criada.

15. Copiar a classe ***TestePessoa*** e dar o nome ***TestePessoa2***.

16. Nesta classe ***TestePessoa2,*** alterar o tipo da instância de ***Pessoa*** para ***Pessoa2*** e testar o programa para tratar uma exceção da nova classe ***ArgumentoForaDosLimitesException***.

17. Criar uma nova classe de exceção **própria *checked***, chamada ***ValorForaDosLimitesException*** derivada da classe ***Exception.***

18. Copiar a classe ***Pessoa*** e dar o nome ***Pessoa3***.

19. Modificar a classe ***Pessoa3*** de modo a usar a nova classe de exceção criada: ***ValorForaDosLimitesException***.

20. Criar uma nova **classe principal** chamada ***TestePessoa3***.

21. Nesta nova classe principal, criar uma instância de ***Pessoa3***.

22. Modificar e testar o programa para visualizar essa instância de ***Pessoa3***.

ArrayLists

1. Criar uma nova classe principal de nome ***TesteArrayList***.
2. Criar algumas **instâncias** da classe ***Pessoa***.
3. Criar um **contentor** do tipo ***ArrayList*** usando o construtor vazio.
4. Visualizar o **tamanho** do contentor.
5. **Adicionar** as instâncias de ***Pessoa*** ao contentor.
6. Visualizar novamente o **tamanho** do contentor.
7. Criar e **testar** o método ***listar*** para visualizar os objetos do contentor, através de um **varrimento** do contentor com a repetição ***for***.
8. Criar e **testar** o método ***listar2*** para visualizar os objetos do contentor, através de um **varrimento** do contentor usando a repetição ***foreach***.
9. Criar e **testar** o método ***listar3*** para visualizar apenas os nomes das instâncias de ***Pessoa*** armazenadas no contentor, através de um **varrimento** do contentor usando a repetição ***foreach***.
10. **Substituir** o **último** objeto do contentor por uma **nova** instância de ***Pessoa***.
11. **Remover** do contentor o último elemento e **visualizar** este elemento.
12. **Tentar remover** do contentor um elemento usando um **índice fora dos limites** (tamanho do contentor).
13. **Adicionar** ao contentor, **entre** objetos guardados, o elemento removido na alínea 11.
14. **Visualizar** o contentor.
15. **Testar** se o objeto adicionado na alínea 13 está **armazenado** no contentor.
16. Criar um **clone** do contentor através do construtor de cópia da classe ***ArrayList***.
17. **Testar** se os dois contentores são **equivalentes** (mesma sequência de objetos).
18. **Remover** um objeto do contentor 2 e visualizar este contentor antes e depois da remoção.
19. **Testar** novamente se os dois contentores são **equivalentes** (mesma sequência de objetos).
20. **Testar** se o contentor 1 **possui** todos os objetos do contentor 2.
21. **Remover todos** os objetos armazenados no contentor 2.
22. **Testar** se este contentor 2 está **vazio**.