INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
IFBA – Campus Irecê
Matéria: Linguagem técnica II
Professor: Wild Barreto

Lista de Exercícios I
Orientação a Objetos

1-) Classe `Retângulo.`

- **Crie uma classe `Retangulo` com atributos `largura` e `altura`.**
- **Implemente métodos para calcular e retornar a área e o perímetro.**
- **Escreva um programa que crie dois retângulos, peça os valores de largura/altura ao usuário e exiba resultados.**

2-) Classe `ContaBancaria`

- **Modele uma conta bancária com atributos numero, titular e saldo.**
- **Métodos: depositar(double valor), sacar(double valor) (não permitir saldo negativo) e mostrarDados().**
- **No main, simule operações de depósito e saque, exibindo o saldo antes e depois.**

3-)Uso de Construtores e `this`

- **Crie a classe `Aluno` com `nome`, `matrícula` e `curso`.**
- **Implemente dois construtores: um sem parâmetros (inicializa campos com valores padrão) e outro recebendo os três campos.**
- **Utilize `this` para distinguir atributos de parâmetros.**
- **No programa principal, instancie alunos com ambos os construtores e exiba seus dados.**

4-)Sobrecarga de Métodos

- **Crie a classe `Calculadora` com métodos `somar(int a, int b)`, `somar(double a, double b)` e `somar(int[] valores)`.**
- **Teste cada versão passando diferentes parâmetros e exibindo os resultados.**

5-)Override de `toString()` e `equals()`

- **Modele uma classe `Contato` com `nome`, `email` e `telefone`.**
- **Sobrescreva `toString()` para exibir os dados em formato legível.**
- **Sobrescreva `equals()` (e `hashCode()`) para que dois contatos sejam iguais quando tiverem o mesmo e-mail.**
- **Teste criando um `HashSet<Contato>` e tente adicionar contatos duplicados por e-mail.**

6-)Composição de Objetos

- **Crie as classes `Endereco` (`rua`, `numero`, `cidade`, `cep`) e `Pessoa` (`nome`, `idade`, `endereço`).**
- **Em `Pessoa`, use um atributo do tipo `Endereco`.**
- **No `main`, peça ao usuário que informe dados de uma pessoa e seu endereço, instancie os objetos e exiba tudo usando `toString()`.**

7-)Herança: `Funcionario` e Subclasses

- **Crie uma classe base `Funcionario` com atributos `nome`, `cpf` e `salarioBase`, e método `calcularSalario()`.**
- **Crie subclasses `Gerente` (adiciona `bonus`) e `Vendedor` (adiciona `comissao`).**
- **Sobrescreva `calcularSalario()` em cada uma conforme a lógica de bônus/comissão.**
- **No `main`, crie uma lista heterogênea de funcionários e exiba seus salários calculados.**

8-)Polimorfismo com Coleções

- **Usando as classes do exercício anterior, armazene diversos `Funcionario` em um `ArrayList<Funcionario>`.**
- **Itere sobre a lista chamando `calcularSalario()` de forma polimórfica e exibindo o resultado.**

9-)Classes Abstratas e Interfaces

- **Crie uma interface `Imprimivel` com método `imprimir()`.**
- **Faça `Produto` (do básico) e `Funcionario` implementarem `Imprimivel`.**
- **Crie uma classe abstrata `Documento` com atributo `titulo` e método abstrato `abrir()`.**
- **Implemente duas subclasses: `Relatorio` e `Carta`, cada uma com sua lógica em `abrir()`.**

10-)Projeto Prático: Biblioteca

- **Modele as classes `Livro` (com `titulo`, `autor`, `isbn` e `emprestado`), `Usuario` (com `nome`, `id`) e `Biblioteca`.**

- **Em `Biblioteca`, gerencie coleções de livros e usuários, e métodos `emprestarLivro(isbn, usuarioId)`, `devolverLivro(isbn)`, `listarLivrosDisponiveis()`.**

- **Use herança para criar `UsuarioPremium` (sem limites de empréstimo) e `UsuarioComum` (limite de 3 livros).**

- **No programa principal, simule operações de empréstimo e devolução, exibindo relatórios de status.**