Ministério da Educação
Universidade Federal de Itajubá – *Itajubá*

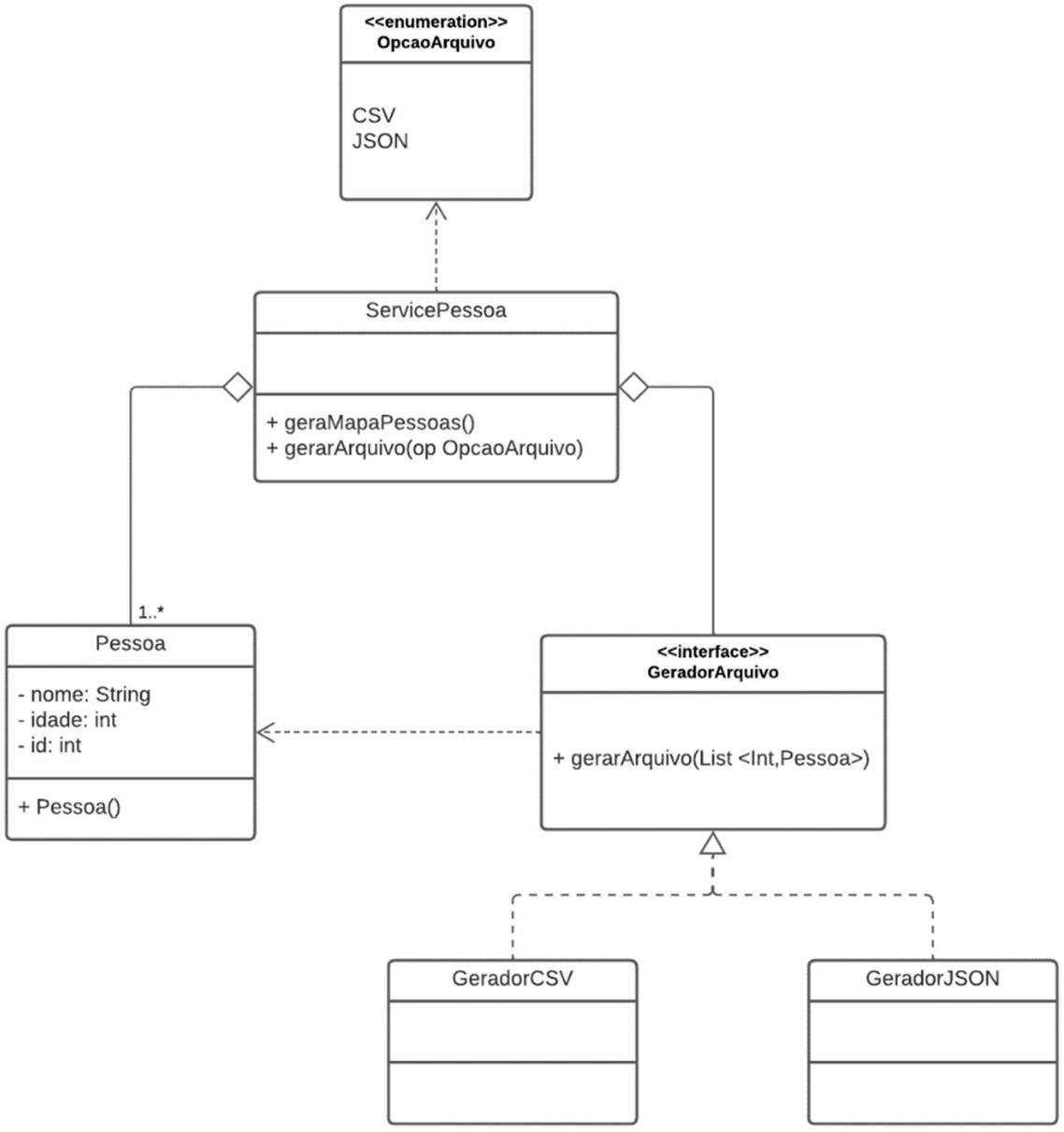
COM221 – Computação Orientada a Objetos II

Exercícios – Aula 3
Prof.: Phyllipe Lima

Instruções Gerais:
- Atividade Individual.
- Entregar a solução completa (código produção e teste) em formato ZIP pela tarefa do SIGAA.
- Data entrega: 09/09 as 18:00
- Faça o download do projeto disponível com os testes (lista-2-referencia). Os testes irão guiar a sua solução.
- Descompacte a pasta e abra no seu ambiente favorito para desenvolver Java. Recomendo IntelliJ.
- Versão mínima com Java 11

Tarefa 1 – Implementando pelo UML

Considere o Diagrama UML abaixo:



Esse diagrama representa um sistema capaz de gerar arquivos, em formato CSV ou JSON, para representar Pessoas. Abaixo temos a classe Pessoa:

```
@Data
@AllArgsConstructor
public class Pessoa {
    private Integer id;
    protected String nome;
    private Integer idade;
}
```

Observe o uso das anotações do ***Lombok***.

A classe ***ServicePessoa*** possui dois métodos. O primeiro, *gerarListaPessoa(),* é um método privado que gera uma lista *hardcoded* com 3 instâncias do tipo Pessoa.

```
void gerarMapaPessoa(){
    pessoas.add(new Pessoa(1,"Guts",39));
    pessoas.add(new Pessoa(2,"Casca",30));
    pessoas.add(new Pessoa(3,"Andreas",60));
}
```

O segundo método, ***gerarAquivo(OpcaoArquivo op),*** recebe qual opção de arquivo se deseja criar, CSV ou JSON. Para minimizar erros, utilizamos um *enum* para ter as opções. Por isso no UML, a classe *ServicePessoa* depende desse *enum.*

Além disso, pelo UML, a classe ***ServicePessoa*** contém uma lista de Pessoas e contém uma referência para a interface ***GeradorArquivo***.

A interface ***GeradorAquivo*** possui um único método, *gerarAquivo(List<Pessoa>),* que possui duas implementações concretas, nas classes ***GeradorCSV*** e ***GeradorJSON.***

A sua tarefa será implementar as funcionalidades para as classes ***GeradorCSV***, ***GeradorJSON*** e ***PessoaService.***
Na classe ***PessoaService,*** o método ***gerarAquivo()*** gera a lista de pessoas, seleciona a opção e redireciona a chamada para umas classes concretas ***GeradorCSV*** ou ***GeradorJSON.***

Verifique se os arquivos “pessoas.csv” e “pessoas.json” foram criados corretamente. Eles foram fornecidos na tarefa para guiar.

Para gerar o CSV, use a biblioteca OpenCSV. Para gerar o JSON, use a biblioteca do Google Gson.

Abaixo um exemplo do método ***main***:

```
public static void main(String[] args) {

    PessoaService p = new PessoaService();
    p.gerarArquivo(OpcaoArquivo.CSV);
    p.gerarArquivo(OpcaoArquivo.JSON);

}
```

Tarefa 2 – Validando e Guiando por Testes de Unidade

Sabemos a importância de testes de unidade para termos segurança com nosso código. Além disso, se tivermos acesso ao código de testes, podemos entender melhor o sistema e até implementá-lo.

No projeto fornecido na tarefa, contém 5 testes. Destes, 3 estão implementados. São os seguintes:

```java
@Test
public void testCriarCSV(){
    Assert.assertTrue(Files.exists(Path.of("pessoas.csv")));
}

@Test
public void testeValidaCSV(){

    try {
        String arquivo = Files.readString(Path.of("pessoas.csv"));
        Assert.assertEquals(ConstFileInfo.CSV,arquivo);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}

@Test(expected = OpcaoErradaException.class)
public void testeOpcaoErrada(){
    service.gerarArquivo(null);
}


@Test //Não implementado
public void testeValidaJSON(){
    Assert.fail();
}

@Test //Não implementado
public void testCriarJSON(){
    Assert.fail();
}
```

A sua tarefa será fazer todos os testes passarem e escrever os dois testes que não foram implementados. Use como exemplo os testes de CSV e o código ***harcoded*** na classe ***ConstFileInfo.***

OBS: Não será permitida nenhuma alteração no código dos testes, com exceção do 2 (dois) testes que você precisará implementar. (testeValidaJSON e testeCriaJSON)