Linguagem de Programação

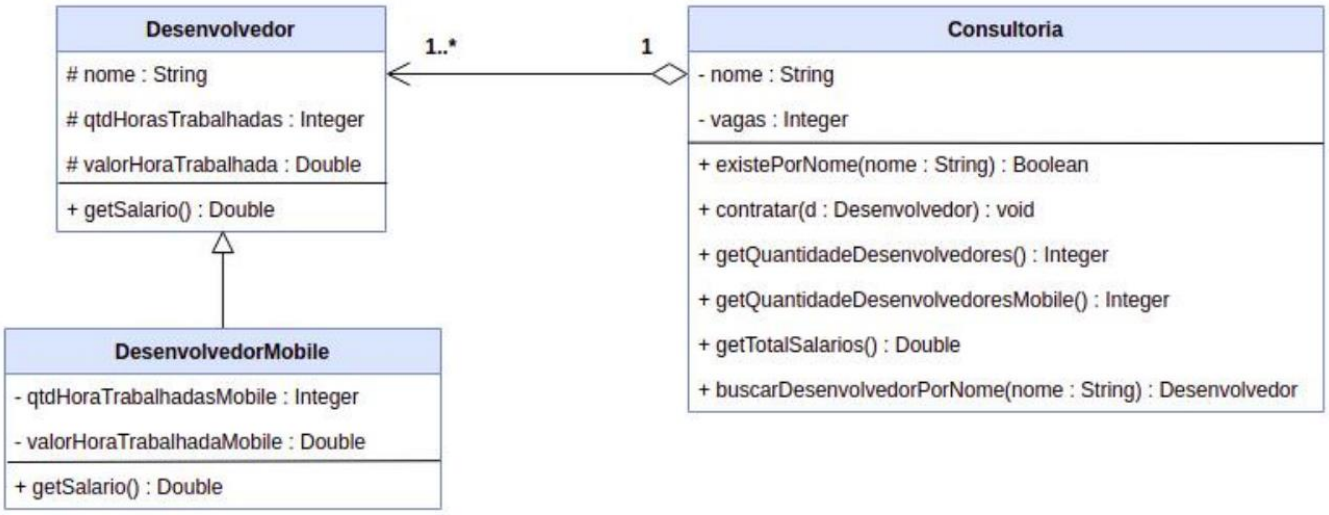
Lista –

Herança com Agregação

Crie um projeto java chamado **heranca-agregacao**

Consultoria:

Crie um programa em Java para controle de remuneração de uma consultoria de desenvolvimento de software, seguindo os requisitos abaixo:



1. Crie uma classe chama `Desenvolvedor`:

   Atributos:

   - `nome`
   - `qtdHorasTrabalhadas`
   - `valorHoraTrabalhada`

   Métodos:

   - `getSalario()`**:** retorna o salário do desenvolvedor (valor da hora*quantidade de horas trabalhadas)
   - `toString()`: exibir os valores dos atributos criados na classe e do cálculodesalário, formato e com uso de interpolação

2. Crie uma classe chama `DesenvolvedorMobile`, sendo uma subclasse de `Desenvolvedor` conforme visto na aplicação dos conceitos de Herança

   Atributos:

   - `qtdHorasTrabalhadasMobile`
   - `valorHorasTrabalhasMobile`

   Métodos:

   - `getSalario()`: sobrescreva o método herdado da superclasse, retorna o salário do desenvolvedor mais o salário do desenvolvedor mobile.
   - `toString()`: exibir os valores dos atributos criados na classe e do cálculo de salário, formato e com uso de interpolação

3. Crie uma classe chama Consultoria

   Atributos:

   - `nome`
   - `vagas`

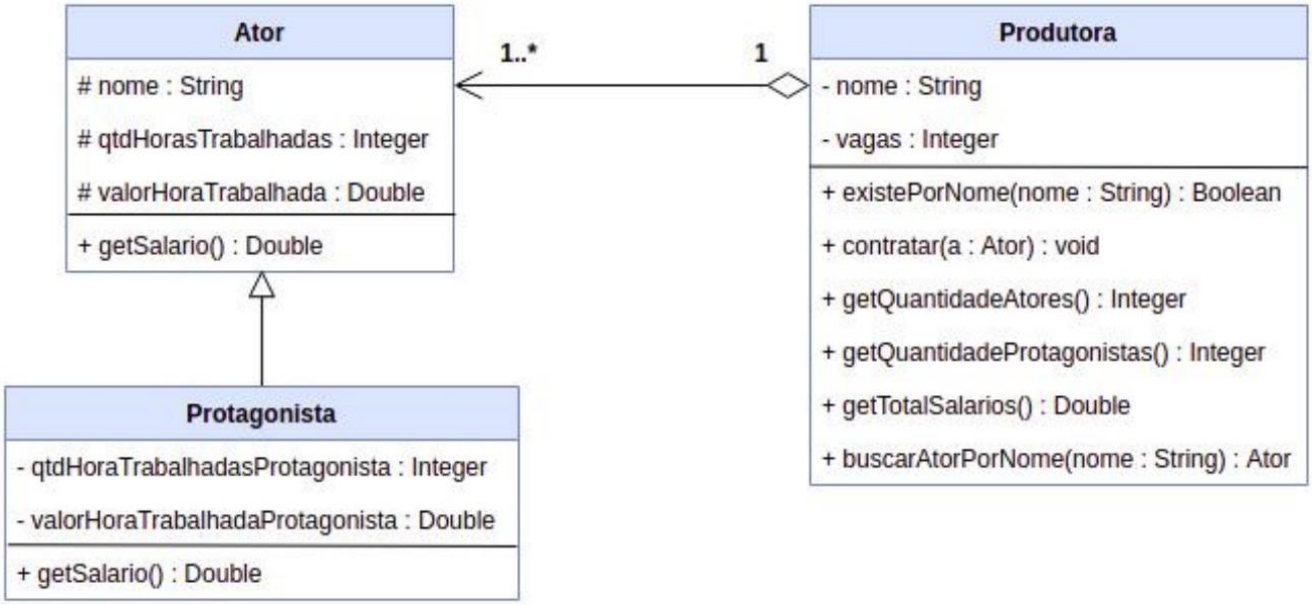
   Métodos:

   - `existePorNome()`: recebe o nome de um desenvolvedor e verifica se ele existe na consultoria, se encontrado retornar `true`, caso contrário retornar `false`
   - `contratar()`: recebe um desenvolvedor e adiciona na consultoria, exiba a frase `"Sem vagas disponíveis!"` caso as vagas estejam esgotadas.
   - `getQuantidadeDesenvolvedores()`: retorna um número inteiro com a quantidade de desenvolvedores na consultoria
   - `getQuantidadeDesenvolvedoresMobile()`: retorna um número inteiro com a quantidade de desenvolvedores mobile na consultoria
   - `getTotalSalarios()`: retorna a soma dos salários de todos os desenvolvedores
   - `buscarDesenvolvedorPorNome()`: recebe um nome e busca na lista de desenvolvedores, caso não encontrado retornar `null`
   - `toString()`: exibir os valores dos atributos criados na classe, formato e com uso de interpolação.

4. Crie uma classe `TesteConsultoria` e que deve conter uma instância de **`Consultoria`** e pelo menos 3 desenvolvedores, testando os métodos desenvolvidos e suas validações.

Produtora

Crie um programa em Java para controle de remuneração de uma produtora de filmes, seguindo os requisitos abaixo:



1. Crie uma classe chama `Ator`

   Atributos:

   - `nome`
   - `qtdHorasTrabalhadas`
   - `valorHoraTrabalhada`

   Métodos:

   - `getSalario()`: retorna o salário do ator (valor da hora * quantidade de horas trabalhadas)
   - `toString()`: exibir os valores dos atributos criados na classe e do cálculo de salário, formato e com uso de interpolação

2. Crie uma classe chama **`Protagonista`**, sendo uma subclasse de **`Ator`** conforme visto na aplicação dos conceitos de Herança

   Atributos:

   - **nome**
   - **qtdHorasTrabalhadasProtagonista**
   - **valorHorasTrabalhasProtagonista**

   Métodos:

   - **`getSalario()`**: sobrescreva o método herdado da superclasse, retorna o salário do ator mais o salário do protagonista.
   - **`toString()`**: exibir os valores dos atributos criados na classe e do cálculo de salário, formato e com uso de interpolação

3. Crie uma classe chama **`Produtora`**

   Atributos:

   - **`nome`**
   - **`vagas`**

   Métodos:

   - **`existePorNome()`**: recebe o nome de um ator e verifica se ele existe na produtora, se encontrado retornar **true**, caso contrário retornar **false**
   - **`contratar()`**: recebe um ator e adiciona na produtora, exiba a frase **`"Sem vagas disponíveis!"`** caso as vagas estejam esgotadas.
   - **`getQuantidadeAtores()`**: retorna um número inteiro coma quantidade de atores na produtora
   - **`getQuantidadeProtagonista()`**: retorna um número inteiro com a quantidade de protagonistas na produtora
   - **`getTotalSalarios()`**: retorna a soma dos salários de todos os atores
   - **`buscarAtorPorNome()`**: recebe um nome e busca na lista de atores, caso não encontrado retornar **null**
   - **toString()**: exibir os valores dos atributos criados na classe, formato e com uso de interpolação

4. Crie uma classe **`TesteProdutora`** e que deve conter uma instância de Produtora e pelo menos 3 atores, testando os métodos desenvolvidos e suas validações.