Lista de Exercícios

Objetivo: Conceitos de Herança e Polimorfismo.

Cenário da Aplicação:

O sistema de uma empresa de manufatura deve possuir um controle de ponto para registrar a entrada e saída dos funcionários. O pagamento dos funcionários depende dessas informações. Desta forma pretende-se criar além das classes bases para aplicação uma classe para implementar o funcionamento de um **relógio de ponto**. Há três tipos de funcionários existentes no sistema: **Gerente, Secretaria e Operador**.

Conteúdo Trabalhado

- Herança e especialização de classes
- Polimorfismo (uso de listas polimórficas)
- Composição (registro de ponto com funcionário associado)
- Manipulação de data e hora com **LocalDate** e **LocalDateTime**
- Encapsulamento e boas práticas de código
- Separação em pacotes e modularidade

Para a resolução do cenário apresentado você deve efetuar os exercícios conforme abaixo:

1) Crie um novo projeto Java chamado **ControlePonto**. A classe que contém o método main deve ser chamada de **GerenciarControlePonto**.

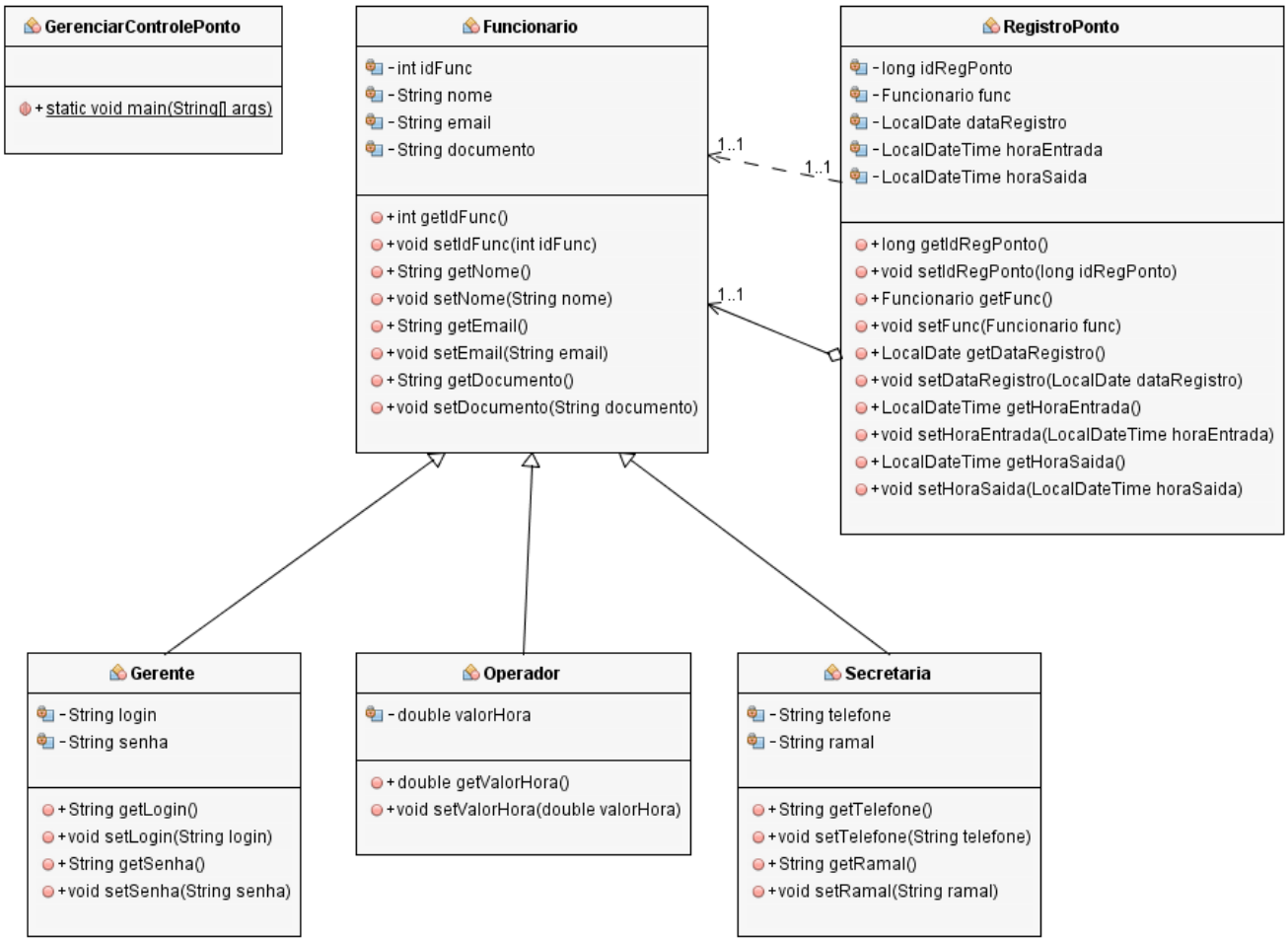
2) Crie uma nova classe Java chamada **Funcionario**, com atributos conforme modelagem na figura a seguir. Os atributos devem estar todos declarados como privados, além disso, defina os métodos getters / setters para prover acesso aos campos.

3) Crie uma nova classe Java chamada **Gerente** que seja uma subclasse de **Funcionario**, com os seguintes atributos: login e senha. Os campos devem ser todos declarados como privados, além disso, defina os métodos getters / setters para prover acesso aos campos.

4) Crie uma nova classe Java chamada **Secretaria** que seja uma subclasse de **Funcionario**, com os seguintes atributos: telefone e ramal. Os campos devem estar definidos como privados, além disso, defina os métodos getters / setters para prover acesso ao campo.

5) Crie uma nova classe Java chamada **Operador** que seja uma subclasse de **Funcionario**, com o seguinte atributo: valorHora. O campo deve estar definido como privado, além disso, defina os métodos getters / setters (conforme padrão Java) para prover acesso ao campo privado.

6) Crie uma nova classe Java chamada **RegistroPonto**, com os seguintes atributos: idRegPonto, func do tipo funcionário, dataRegistro do tipo LocalDate, horaEntrada do tipo LocalDateTime e horaSaida do tipo LocalDateTime. Os campos devem estar definidos como privados, além disso, defina os métodos getters / setters para prover acesso ao campo.



Lembrando: Além de gerar reaproveitamento de código, a utilização de herança permite que objetos criados a partir das classes específicas sejam tratados como objetos da classe genérica. Em outras palavras, a herança entre as classes que modelam os funcionários permite que os objetos criados a partir das classes: **Gerente**, **Secretaria** ou **Operador** sejam tratados como objetos da classe **Funcionario**. Característica essa, conhecida por polimorfismo.

Analisando:

O registro do ponto na empresa não depende do cargo do funcionário. Não faz sentido criar um método que registre a entrada para cada tipo de funcionário, pois eles serão sempre idênticos. Analogamente, não faz sentido criar um método que registre a saída para cada tipo de funcionário. Diante disso, a classe **RegistroPonto** possui os métodos: setDataRegistro, setHoraEntrada e setHoraSaida, esses métodos serão responsáveis por receber os dados de registro de horário dos funcionários independentemente do tipo de funcionário que está entrando ou saindo do local de serviço.

Nota: Use variáveis do tipo java.time.LocalDateTime e java.time.LocalDate. A capacidade de tratar objetos criados a partir das classes específicas como objetos de uma classe genérica é chamada de **polimorfismo**.

7) Na Classe RegistroPonto acrescente o método apresentarRegistroPonto(), que deverá apresentar os dados com o nome do funcionário a data de registro do ponto e os horários de entrada e de saída respectivamente. Pode ser retornar uma String (toString), se preferir.

Na Classe GerenciarControlePonto implemente o método main, de acordo com o pedido.

8) Instancie um **Gerente** e popule os dados.
9) Instancie uma **Secretaria** e popule os dados.
10) Instancie uma **Operador** e popule os dados.

11) Registre a entrada na empresa do Gerente, apresentado no console os dados da entrada. Lembre-se que os métodos estão na classe RegistroPonto. **Nota:** cada lançamento de entrada/saída no dia para o funcionário trata-se de uma nova instância de Registro.
12) Registre a entrada na empresa do Operador, apresentado no console os dados da entrada.
13) Registre a entrada na empresa da Secretaria, apresentado no console os dados da entrada.
14) Registre a saída do Gerente, apresentado no console os dados da saída.
15) Registre a saída da Operador, apresentado no console os dados da saída.
16) Registre a saída da Secretaria, apresentado no console os dados da saída.
17) Acrescente um tempo de um segundo entre cada registro. Pesquise sobre o método **sleep** da classe **Thread**.

Exemplo de Resultado:

```
==================================
Funcionário: João Silva
Data de Registro: 15/04/2025
Hora de Entrada: 17:08:57

==================================
Funcionário: Pedro Santos
Data de Registro: 15/04/2025
Hora de Entrada: 17:08:58

==================================
Funcionário: Maria Oliveira
Data de Registro: 15/04/2025
Hora de Entrada: 17:08:59

==================================
Funcionário: João Silva
Data de Registro: 15/04/2025
Hora de Entrada: 17:08:57
Hora de Saída: 17:09:00

==================================
Funcionário: Pedro Santos
Data de Registro: 15/04/2025
Hora de Entrada: 17:08:58
Hora de Saída: 17:09:01

==================================
Funcionário: Maria Oliveira
Data de Registro: 15/04/2025
Hora de Entrada: 17:08:59
Hora de Saída: 17:09:02

Process finished with exit code 0
```

Bom Trabalho!!

Prof. Maromo