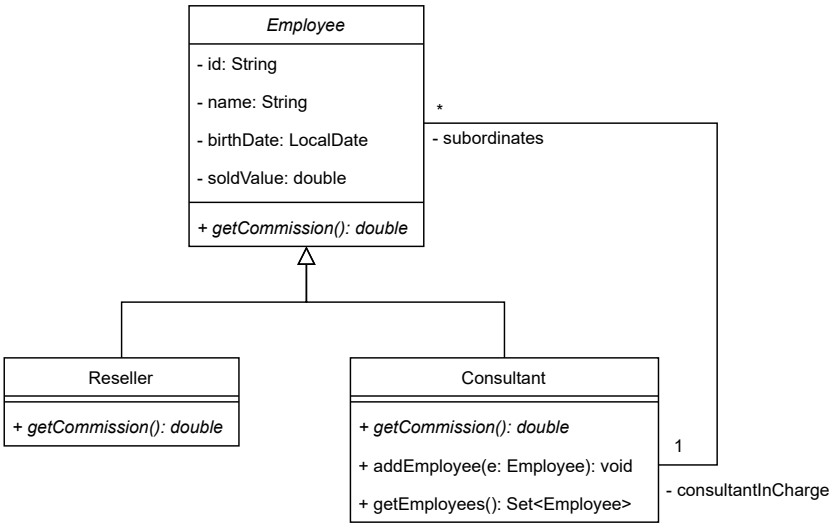
INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Prova 2 – Programação Orientada a Objetos – PROO
Prof. Lucas Oliveira

PRONTUÁRIO: INDICAR NO ZIP Dia: 07/11/2024 Horário: 19h00 – 22h30

✓ A prova é individual e sem consulta. ✓ Não é permitido utilizar qualquer código implementado por você anteriormente. ✓ Não coloque seu nome no projeto. ✓ Atribui-se nota zero à prova em desacordo com os itens acima.	✓ A prova deve ser nomeada da seguinte forma: PRONTUARIO_P2, com o "SC". ✓ Não envie apenas as classes, mas todo o projeto. ✓ Envie o projeto como zip no Moodle.

Descrição do contexto – Algumas empresas com Natura e HerbaLife possuem um sistema de vendas que funciona por meio de consultores. Um Revendedor (*Reseller*), ao atingir um grande número em vendas, pode empregar outros funcionários para que eles o auxiliem, tornando-se um Consultor (*Consultant*). Um Funcionário (*Employee*) pode ser tanto um simples revendedor quanto um consultor. Revendedores ganham 15% do valor bruto de suas vendas. Consultores ganham, além das porcentagens de suas vendas, o equivalente a 30% do valor total das comissões recebidas pelos funcionários sob sua imediata responsabilidade. O cálculo do valor dos ganhos obtidos por consultores e revendedores de uma empresa demanda muito tempo e muitas vezes resulta em erros. Por isso, você precisa criar uma aplicação que calcula os valores dos ganhos de todos os funcionários da empresa. O diagrama de classes do modelo da aplicação é apresentado a seguir:



Dados para teste [id, name, birthDate, soldValue, consultantInChargeId]:

- "12312312312", "David A. Huffman", LocalDate.parse("1925-08-09"), 7000.0, null)
- "32132132131", "Augusta Ada Byron", LocalDate.parse("1852-11-27"), 3000.0, "12312312312"
- "21321321313", "Edsger Wybe Dijkstra", LocalDate.parse("1930-05-11"), 1520.0, "12312312312"
- "45645645646", "Alan Mathison Turing", LocalDate.parse("1912-06-23"), 780.0, "32132132131"
- "90219021902", "Donald Ervin Knuth", LocalDate.parse("1938-01-10"), 432.0, "45645645646"
- "54654654654", "Grace Murray Hopper", LocalDate.parse("1906-12-09"), 432.0, "21321321313"
- "65465465464", "John von Neumann", LocalDate.parse("1903-12-28"), 300.00, "45645645646"

Execute as atividades a seguir para a implementação do exercício em Java. Para a atribuição da nota será levada em conta não apenas a funcionalidade, mas a qualidade, adequação e pertinência de cada solução. Bom senso faz parte da prova.

Dica 1: Antes de começar a implementar, desenhe os relacionamentos entre os dados de teste e calcule manualmente as comissões. Assim, você entenderá melhor o problema.
Dica 2: Use sistematicamente a funcionalidade de *debugging* do IntelliJ para verificar se as suas implementações estão produzindo os resultados esperados.

#	Descrição	Pont.
1	Crie o projeto, os arquivos representando as classes do modelo e um arquivo adicional chamado Main, que contenha o método main do projeto. Crie pacotes chamados *model*, *service*, *persistence* e *exception*. Aloque as classes do diagrama no pacote *model*.	0,5pt
2	Implemente a classe Employee, fazendo com que ela controle quais classes podem a herdar. Faça com que as subclasses de Employee sejam obrigadas a sobrescrever o método getCommission(). Adote o princípio do encapsulamento e sobrescreva métodos de Object que possibilitem a representação do objeto como String, bem como seu adequado uso em comparações, coleções e mapas.	1,5pt
3	Implemente as classes Consultant e Reseller, sobrescrevendo o método previsto na superclasse, de acordo com a especificação. Faça com que estas classes não possam ser herdadas. Utilize sets para representar relacionamentos do tipo um para muitos. Dica: no método addSubordinate() de Consultant, tenha cuidado para não adicionar funcionários em duplicidade caso eles tenham sido promovidos (ver Item 7).	1,5pt
4	Crie uma interface chamada Repository dentro do pacote *persistence*. Adicione métodos save, update e findById. Para obter a pontuação completa na questão, faça com que a interface seja genérica em termos do tipo da chave e do objeto persistido, bem como previna que métodos de consulta venham a retornar objetos nulos.	1,0pt
5	Crie uma exceção não verificada chamada EntityAlreadyExistsException e a aloque no pacote *exception*.	0,5pt
6	Crie uma classe chamada InMemoryEmployeeRepository, que implemente a interface Repository para armazenar objetos Employee na memória. Para emular o banco de dados, utilize uma estrutura de mapa, que seja única na aplicação. O método save não deve permitir a inserção de funcionários com o mesmo id. Neste caso, dispare uma EntityAlreadyExistsException. O método findById() não deve obter apenas o elemento referente ao id, mas todas as associações de subordinados alcançáveis a partir deste objeto.	2,0pts
7	Crie uma classe chamada EmployeeRegistrationService, que receba por injeção no construtor uma dependência de objeto Repository. Aloque esta classe no pacote *service*. Crie um método público chamado register, que receba os dados para a criação de um novo revendedor no repositório, incluindo o id do empregado responsável por ele. Caso o Employee responsável seja um revendedor, ele precisará ser promovido a consultor.	1,5pt
8	Crie uma classe chamada EmployeeReportService, que receba por injeção no construtor uma dependência de objeto Repository. Aloque esta classe no pacote *service*. Crie um método chamado reportOf(), que receba um id e retorne uma String contendo os dados e valor de comissão do funcionário referente ao id e todos os funcionários sob sua supervisão. Para melhor visualização da hierarquia da empresa, adote o formato do Apêndice A.	1,5pt
9	No método Main, crie um repositório concreto e o injete nas classes de serviço. Na sequência, crie objetos para os dados de teste e os registre com o apoio da respectiva classe de serviço. Também utilizando uma classe de serviço, produza um relatório para o funcionário no topo da hierarquia de vendas e, depois, realize a impressão no console.	1,0pt

O aluno que terminar todas as atividades corretamente primeiro ganha 1,0pt adicional para usar em outros exercícios.

***** Boa sorte! ****

APÊNDICE A

[12312312312] David A. Huffman | Birthday: 1925-08-09 | Amount in sales: US$7000.00 | Commission: US$1272.73

[21321321313] Edsger Wybe Dijkstra | Birthday: 1930-05-11 | Amount in sales: US$1520.00 | Commission: US$247.44

[54654654654] Grace Murray Hopper | Birthday: 1906-12-09 | Amount in sales: US$432.00 | Commission: US$64.80

[32132132131] Augusta Ada Byron | Birthday: 1852-11-27 | Amount in sales: US$3000.00 | Commission: US$494.98

[45645645646] Alan Mathison Turing | Birthday: 1912-06-23 | Amount in sales: US$780.00 | Commission: US$149.94

[65465465464] John von Neumann | Birthday: 1903-12-28 | Amount in sales: US$300.00 | Commission: US$45.00

[90219021902] Donald Ervin Knuth | Birthday: 1938-01-10 | Amount in sales: US$432.00 | Commission: US$64.80