LISTA DE EXERCÍCIOS

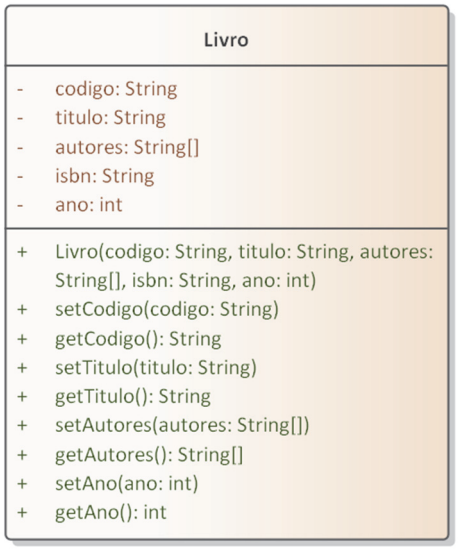
LISTA	TEMA
04	**Orientação a objetos**

OBJETIVOS

- Praticar o uso de classes e relacionamento entre objetos.
- Utilizar Encapsulamento, Associação e Herança na linguagem Java.

EXERCÍCIOS

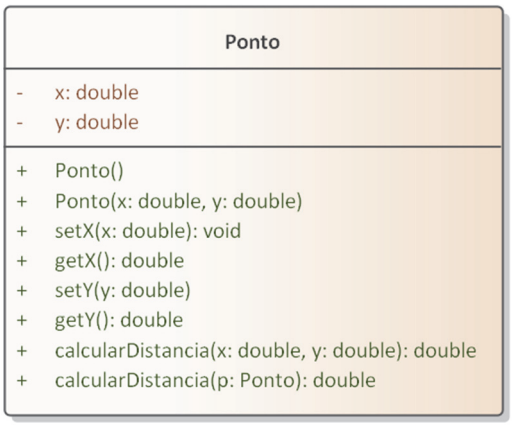
Cenário 1: Crie uma classe que represente um Livro de uma biblioteca. Em outra classe (Programa), instancie os seguintes livros e apresente seus dados na tela. (Não precisa usar Model-View-Controller MVC)



Livro 01:
Código: 1598FHK
Título: Core Java 2
Autores: Cay S. Horstmann e Gary Cornell
ISBN: 0130819336
Ano: 2005

Livro 02:
Código: 9865PLO
Título: Java, Como programar
Autores: Harvey Deitel
ISBN: 0130341517
Ano: 2015

Cenário 2: Crie uma classe que represente um Ponto no espaço bidimensional. Na classe Programa efetue as operações listadas abaixo. Para realização dos cálculos de elevação ao quadrado utilizar **Math.pow(valor, 2)** e para extração da raiz quadrada utilizar **Math.sqrt(valor)**. (Não precisa usar Model-View-Controller MVC)



1. Crie um objeto ponto1 usando o primeiro construtor;
2. Crie um objeto ponto2 na posição (2,5);
3. Calcule a distância do ponto1 ao ponto2;
4. Calcule a distância do ponto2 às coordenadas (7,2);
5. Altere o valor de x para 10 no ponto1;
6. Altere o valor de y para 3 no ponto1;

Observações:

- O construtor Ponto( ): cria um ponto na origem (0,0);
- O Ponto(x: double, y: double): nas coordenadas x e y;

- Cálculo da distância entre dois pontos:

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

Cenário 3: Agenda de Compromissos (Não precisa usar Model-View-Controller MVC)

A agenda deve permitir a criação de um compromisso com uma pessoa, em um determinado local e horário, para tratar um assunto específico. Para cada dia considerar que será possível agendar o compromisso para qualquer uma das 24 horas de um dia, qualquer dia do mês de 28 a 31 dias e em qualquer um dos 12 meses do ano.

Cenário 4: Controle de estacionamento

Um estacionamento precisa controlar o fluxo de veículos que entram e saem diariamente de seu espaço. Para isso, seu operador precisa de um sistema que registre as informações de modelo, placa e cor dos carros toda vez que eles chegam. Ao entrar no estacionamento o carro ocupa uma das 10 vagas disponíveis, portanto é necessário verificar se o estacionamento ainda dispõe de vagas e na retirada é necessário buscar o veículo através de sua placa. No final de cada período, manhã, tarde ou noite, é necessário emitir um relatório que informe quantos veículos entraram e saíram do estacionamento e o valor de pagamentos naquele período, considerando o valor do por período é de R$ 5,00. O sistema deve ser desenvolvido no padrão arquitetural Model-View-Controller (MVC), possuir uma classe Programa que inicia a execução do software e um menu que permita ao usuário realizar as operações de entrada e saída de veículos.

Cenário 5: Controle de eventos

Uma empresa de eventos precisa de um sistema que lhe permita gerenciar eventos e reservas. Para cada evento ela precisa registrar nome, data, local, lotação máxima do evento, quantidade de ingressos vendidos e o preço do ingresso. Os clientes podem realizar a reserva para o evento informando nome do responsável pela reserva e quantidade de pessoas. Na reserva ainda deve ser registrado a data e o valor total da reserva. O sistema deve ser desenvolvido no padrão arquitetural Model-View-Controller (MVC), possuir uma classe Programa que inicia a execução do software e um menu que permita ao usuário realizar as operações de inclusão, alteração, listagem e exclusão de eventos e reservas.

Cenário 6: Controle de tráfego aéreo

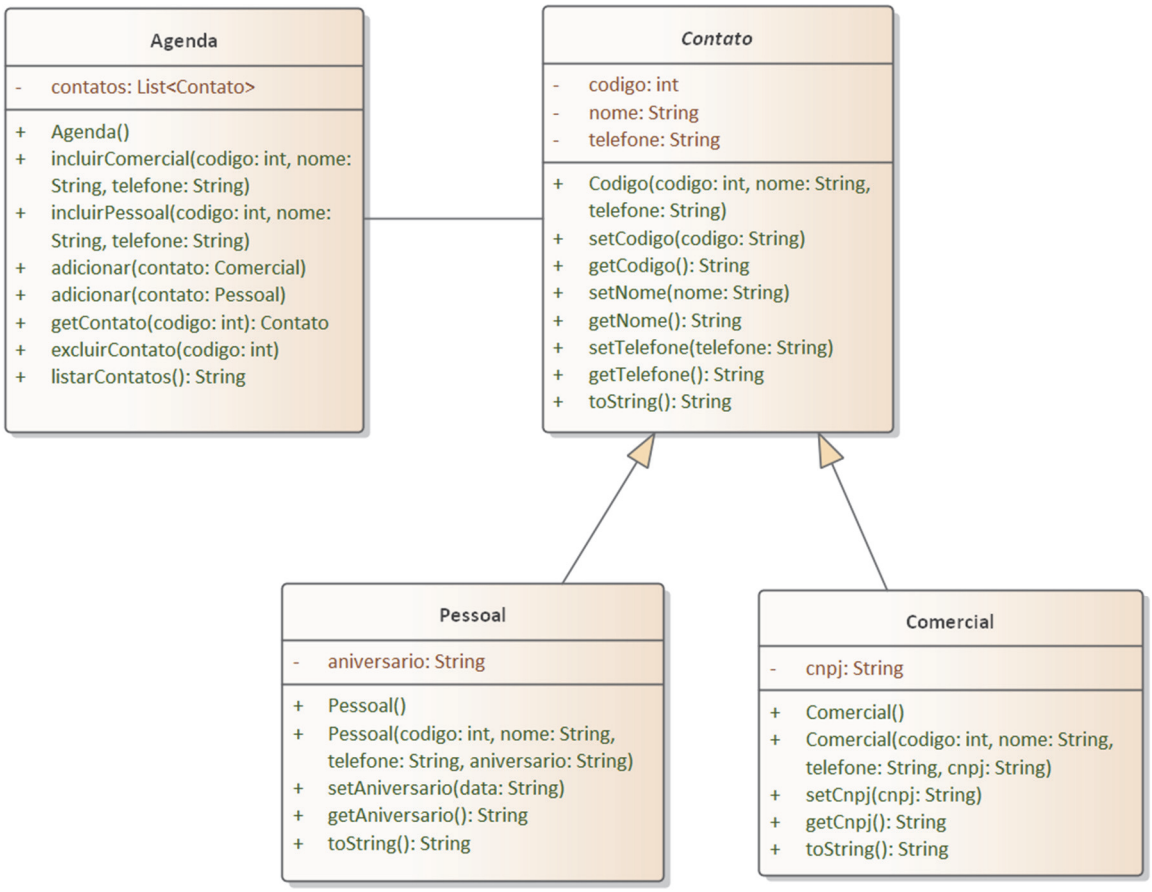
Um aeroporto precisa-se registrar informações sobre as diferentes pessoas que nele trafegam. Todas as pessoas possuem nome e rg. As pessoas se dividem entre passageiros e tripulação. Os passageiros possuem um identificador de bagagem e a sua passagem. Sobre a passagem, armazena-se o número do acento, a classe do acento e a data do voo, contendo dia, mês, ano, hora e minuto de partida. Sobre a tripulação, sabe-se a sua identificação aeronáutica e matrícula do funcionário. Dos comandantes, registra-se o seu total de horas de voo e dos comissários os idiomas em que possuem fluência. Todas as pessoas possuem ainda informações sobre a aeronave em que farão o voo. Sobre ela, armazena-se o seu código, tipo e quantidade de assentos. O sistema deve ser desenvolvido no padrão arquitetural Model-View-Controller (MVC), possuir uma classe Programa que inicia a execução do software e um menu que permita ao usuário realizar as operações necessários para o controle do tráfego.

Cenário 7: Controle acadêmico

Em uma instituição de ensino, devem ser registradas informações sobre professores, alunos e seus relacionamentos entre disciplinas. Todas as pessoas representadas no sistema possuem nome, rg e matrícula. O software deve ser desenvolvido no padrão arquitetural Model-View-Controller (MVC). Os professores possuem número de identificação do seu currículo Lattes e titulação, envolvendo nome da instituição, ano de conclusão, nome do título obtido e título do trabalho de conclusão. Os alunos, por sua vez, possuem o ano de ingresso na instituição, nome do curso e turno. Todas as disciplinas possuem um nome, identificador, currículo a que pertencem e um conjunto de competências, classificadas como Necessárias e Complementares. Na disciplina também estão registrados o professor que a ministra e os alunos nela matriculados. Para cada aluno é registrada a situação acerca de suas competências, sendo ela Atingida ou Não-Atingida. A partir da sua situação, pode-se avaliar a situação do aluno como:

- Aprovado: 100% das competências Necessárias, pelo menos 50% das competências complementares;
- Reprovado: menos de 50% das competências Necessárias ou menos de 50% das competências complementares;
- Pendente: nenhuma das duas situações anteriores.

Cenário 8: Agenda de contatos (classe Contato abstrata)

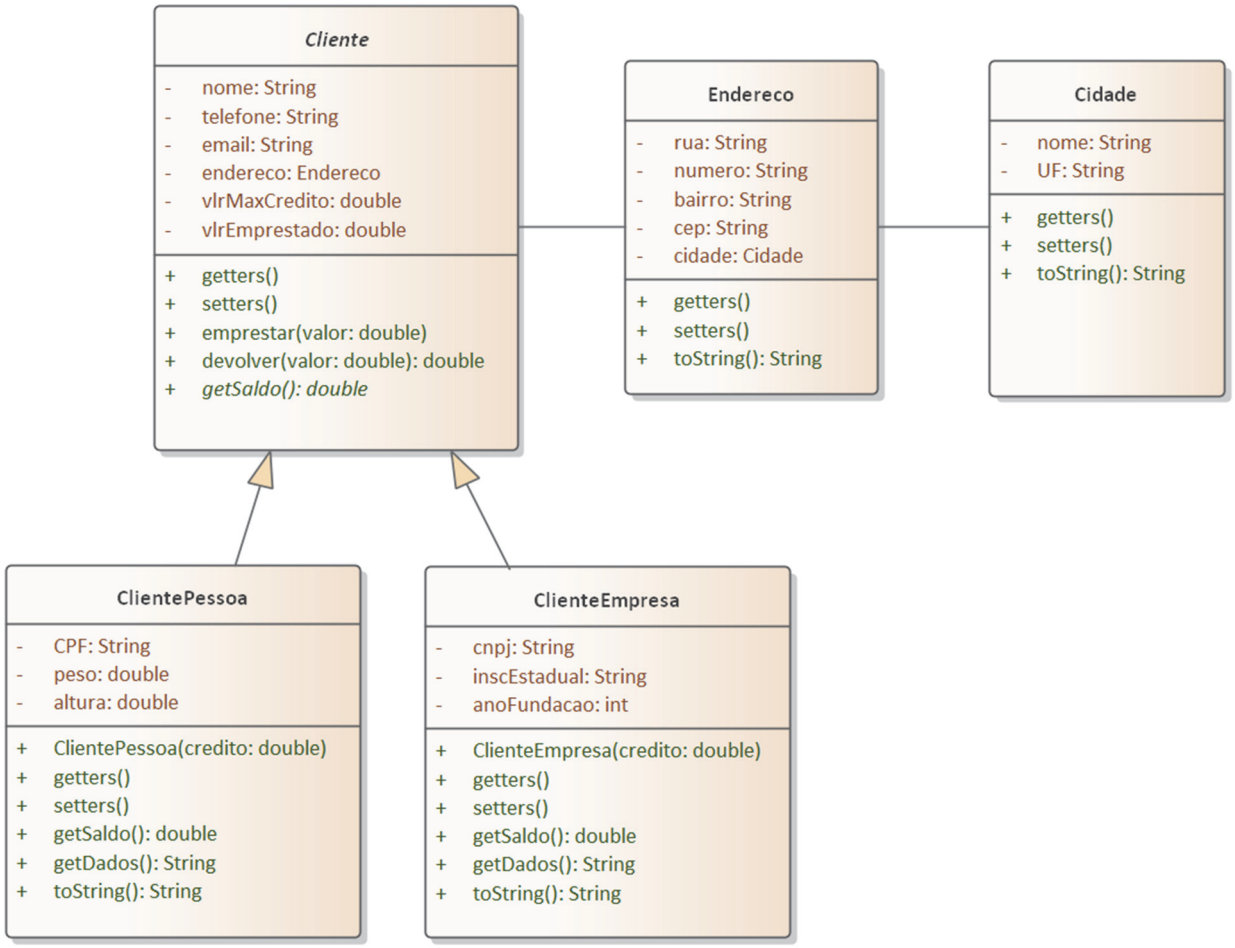


Para a execução do software será necessário criar a classe Programa e utilizar o padrão Model-View-Controller (MVC) para organizar a arquitetura do software. A classe Contato é abstrata (não permite a geração de objetos). Os valores passados aos objetos criados pelas classes Pessoal e Comercial devem ser lidos do teclado.

De acordo com as classes implementadas, implemente uma agenda de contatos com um menu onde é permitido:

1. Incluir um contato pessoal
2. Incluir um contato comercial
3. Excluir um contato pelo código
4. Consultar um contato pelo código
5. Listar todos os contatos
6. Sair do programa

Cenário 9: Cadastro de clientes (classe Cliente abstrata)



Tendo em vista o diagrama apresentado, desenvolva uma aplicação utilizando o padrão Model-View-Controller (MVC) para organizar a arquitetura do software e implemente os seguintes tópicos:

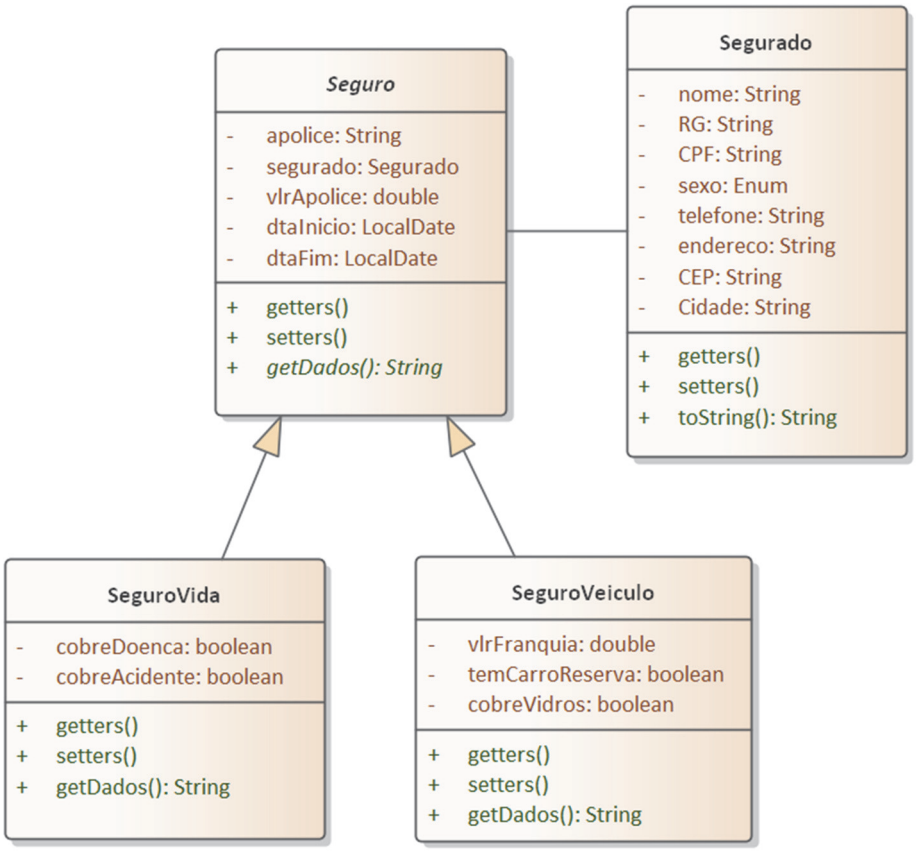
- As classes ClientePessoa e ClienteEmpresa herdam atributos e métodos da classe Cliente (abstrata e não permite a geração de objetos);
- As classes devem possuir os getters e setters para seus atributos assim como os demais métodos identificados para cada uma delas, respeitando as assinaturas;
- O atributo vlrEmprestado deve ser manipulado pelos métodos emprestar() e devolver() considerando que os métodos devem ser responsáveis por impedir que o valor emprestado exceda o valor máximo de crédito ou fique negativo;
- O valor máximo de crédito para cliente pessoa é de R$ 10.000,00 e cliente empresa R$ 25.000,00.

<u>O menu apresentado ao usuário deve conter os seguintes itens (encerrar o programa no Sair):</u>

1. Incluir cliente pessoa
2. Incluir cliente empresa
3. Mostrar dados cliente pessoa
4. Mostrar dados cliente empresa
5. Emprestar para cliente pessoa

6. Emprestar para cliente empresa
7. Devolução de cliente pessoa
8. Devolução de cliente empresa
9. Sair

Cenário 10: Apólice de seguros (classe Seguro abstrata)



Desenvolva um software utilizando o padrão arquitetura Model-View-Controller (MVC) e implemente o digrama de classes considerando os seguintes tópicos:

- Deve ser criado um projeto separado para a implementação;
- As classes SeguroVida e SeguroVeiculo herdam da classe Seguro (abstrata);
- As classes devem possuir os getters e setters para seus atributos assim como os demais métodos, respeitando as assinaturas (quando estiver completa);
- Podem ser implementados métodos e atributos para as classes além dos especificados no diagrama;
- Quando o usuário informar o número da apólice, o sistema deve verificar se ele já não foi inserido, avisando o usuário imediatamente, evitando que sejam digitados todos os dados;
- Deverá ser implementado um método excluirTodosSeguros que deve pedir confirmação antes de excluir todos os seguros do vetor;
- Um outro método quantidadeSeguros deve retornar quantos seguros estão inseridos;
- O menu apresentado ao usuário deve conter os seguintes itens (encerrar o programa ao Sair):

1. Incluir seguro
2. Localizar seguro
3. Excluir seguro
4. Excluir todos os seguros
5. Listar todos os seguros
6. Ver quantidade de seguros
7. Sair