Trabalho Prático
Disciplina: Laboratório de Engenharia de Software I

A nota do trabalho será divida em 2 fases:
Fase 1: 40 pontos
Fase 2: 40 pontos

Neste trabalho prático, a turma deverá criar um sistema de reserva de hotéis, passagens aéreas e aluguel de carro. Para isso, na primeira fase, este sistema será divido para cada grupo fazer uma parte e, na segunda fase, cada grupo se juntará com mais 1 ou 2 grupos para juntar e integrar suas implementações em apenas uma – mais completa.

Na primeira fase, cada grupo irá implementar algumas classes do sistema. Dentre as classes a ser implementadas, algumas terão seus objetos criados manualmente e outras os dados deverão ser requisitados para o usuário. Ao solicitar para o usuário, teremos que possibilitar ao usuário que liste, crie, altere e remova instâncias de uma classe, isso é chamado de CRUD (do inglês, *Create, Read, Update and Delete*). Por exemplo, num sistema de reservas de voos, é necessário criar, listar, atualizar e remover voos. Os grupos deverão implementar exatamente o que está descrito neste documento e poderão adicionar métodos e atributos que julgarem necessário (mas, evitando redundância de dados).

Este documento detalha o que cada grupo deverá implementar em cada uma das duas fases do projeto. São exemplos de requisitos que devem ser atendidos:

- Cada grupo poderá escolher a tecnologia de implementação;
- Os objetos instanciados pelo usuário deverão ser armazenados em tabelas do banco de dados;
- Usar comentários ao longo do código e evitar os *bad smells* no código vistos em sala de aula;

Definição dos grupos

Deverão ser **obrigatoriamente** 8 grupos, divididos da seguinte forma:
- 6 grupos de 4 pessoas;
- 1 grupo de 3 pessoas;
- 1 grupo de 2 pessoas;

É **obrigatório** que os integrantes do grupo sejam da mesma prática. Como são duas turmas de laboratório, a organização dos grupos deverá ser obedecida da seguinte maneira para fins de organização:

Turma 01 (composta por 20 pessoas):
- 5 grupos de 4 pessoas

Turma 02 (composta por 9 pessoas)
- 1 grupo de 4 pessoas
- 1 grupo de 3 pessoas
- 1 grupo de 2 pessoas

Prazos

- Definição dos grupos: **até 30/09**
- Entrega da primeira fase: **até 6/11**
- Arguição dos grupos: **13/11**
- Entrega da segunda fase: **4/12**

Cada dia de atraso em qualquer entrega acarretará em **perda de 2 pontos**.

Arguição

A arguição é obrigatória para todos os alunos do grupo. Assim, serão feitas perguntas para cada integrante sobre a primeira fase e andamento da segunda e o mesmo será avaliado de acordo com o seu conhecimento do trabalho. Por isso, cada aluno deverá conhecer completamente o funcionamento do código que o grupo implementou, pois cada integrante será avaliado **individualmente** e a sua nota final ficará definida de acordo com a performance nesta entrevista. Caso nenhum integrante consiga responder as perguntas, o grupo ficará com zero no trabalho. Caso um aluno falte, ele perderá todos os pontos do trabalho.

Forma de avaliação:

O trabalho será avaliado (tanto a primeira quanto a segunda fase). A partir dos seguintes critérios objetivos:

Critérios de Correção	**Nota**
O não comparecimento à arguição do trabalho	-10 pontos
Trabalhos que não compilarem	-10 pontos
Trabalhos que compilarem, mas gerarem resultados incorretos	Até -7 pontos, dependendo do número de erros
Entregues e definição do grupo fora do prazo	**Definição dos grupos:** -1 por dia de atraso **Primeira fase:** Até dois dias de atraso: -1 pontos por dia, -2 pontos nos dias subsequentes. Não será aceito trabalho após o quarto dia de atraso. **Segunda fase:** -1 ponto por dia de atraso. Não será aceito o trabalho após o terceiro dia de atraso.
Caso haja mais integrantes no grupo que permitido	-2 pontos por integrante excedente

Arguição: a nota final será individual proporcional à performance de cada aluno na arguição.
Além disso, os trabalhos serão avaliados segundo critérios subjetivos definidos pelo professor. Os critérios subjetivos avaliados serão:

- Uso dos princípios básicos da orientação a objetos vistos em sala de aula evitando variáveis globais e excesso de métodos static, evitar passagem de parâmetro quando o objeto já possui determinada informação;
- Legibilidade e Organização (endentação, nomes de variáveis bem escolhidos, código bem formatado, uso de comentários quando necessário, etc.);
- Consistência (utilização de um mesmo padrão de código);
- Entendimento individual a respeito do código-fonte apresentado.

OBS.: Reclamações de notas devem ser feitas com até três dias após o seu lançamento.
Qualquer plágio, ou seja, cópia parcial ou total do trabalho não será tolerada. Caso identificado, ambos os grupos (o que forneceu o código e o que utilizou o código) perderão seus pontos. Alguns grupos, na primeira fase, farão funcionalidades iguais, porém, cada grupo deverá fazer sua própria solução para o problema. Caso tenha ficado evidente que um grupo usou o código de outro grupo parcialmente ou totalmente, será considerado plágio. Por isso, não compartilhe o seu código.

Entregáveis por fase:

Primeira fase: Código-fonte comprimido em um ".zip" e jar executável.

Segunda fase: Código-fonte comprimido em um ".zip" e jar executável (um apenas para todos os grupos). Além disso, deve ter um *readme* com os integrantes de cada grupo que participou da entrega.

Enviar para o e-mail edu@cefetmg.br com o assunto: "TrabalhoES-GrupoID", onde GrupoID é o número identificador do grupo (ex. Grupo 1, Grupo 7, etc.).

Primeira Fase

Veja o diagrama de classe em "sistema_reserva.pdf". Cada grupo fará uma parte deste diagrama.

Grupo 1: CRUD de CiaAereas, Voo, PassagemAerea. Cada Voo deverá ser disponibilizado n passagens aéreas sendo que esta quantidade deve ser menor do que a capacidade da aeronave. Tanto as aeronaves, assento quanto a cidades a serem selecionadas deverão ser inseridas manualmente (sem uso de tela). Não é necessário associar a Pessoa a passagem aérea. Pois, essa pessoa seria a pessoa que comprou a passagem aérea (isso será feito ao efetuar a reserva). O TipoDeTrecho poderá assumir dois valores: Ida ou Volta (usar classe **enumeration**, ver explicação de **enumeration** no **Apêndice**).

Grupo 2: CRUD de Cidade, Estado, Pais e inserção/atualização de Endereço: Existem países que possuem estados e outros que são apenas um conjunto de cidades (sem divisão por estados). No caso de não existir estados, a cidade será associada diretamente ao país, caso contrário, a cidade será associada ao estado correspondente. Todo estado é associado ao país. Logo após, será criado uma tela para criação do endereço, esta tela deverá criar um objeto endereço com seus atributos e cidade correspondente ou, a partir de um objeto endereço, alterá-lo.

Grupo 3: CRUD de Assento, Aeronave, Voo e Passagem Aérea. Cada Voo deverá ser disponibilizado n passagens aéreas sendo que esta quantidade deve ser menor do que a capacidade da aeronave. As cidades a serem selecionadas deverão ser inseridas manualmente (sem uso de tela). Não é necessário associar a Pessoa a passagem aérea. Pois, essa pessoa seria a pessoa que comprou a passagem aérea (isso será feito ao efetuar a reserva). O TipoDeTrecho poderá assumir dois valores: Ida ou Volta (usar classe **enumeration**, ver explicação de **enumeration** no final da especificação).

Grupo 4: CRUD de Pessoa, Cliente e de itens recomendáveis (Classes que herdam de Recomendavel). O CRUD de cliente, deverá inserir todos os dados da pessoa também. Pode-se recomendar hotéis e passagens aéreas para clientes. A recomendação pode ser feita de duas formas diferentes: haverá um cadastro prévio de itens recomendáveis: uma tela para recomendar quartos de hotéis e outra para recomendar passagens aéreas. Tanto hotéis quanto passagens aéreas devem ser cadastradas manualmente, ou seja, não há necessidade de criar telas para cadastro de hotéis e de passagens aéreas a tela de recomendação apenas fará associação entre o item recomendável ao cliente. Além disso, na tela de lista de clientes, ao selecionar um cliente, deve-se exibir as recomendações para este cliente. Os dados a serem exibidos são a descrição desta recomendação por meio do método getDescricaoRec().

Grupo 5: CRUD de ReservaAereo e associação à PassagemAerea: Em uma ReservaAereo, o usuário deve definir a (1) quantidade de pessoas (2) cidade origem e (2) cidade destino a data de ida e data de volta (se for viagem de ida e volta). Assim será exibido duas listas: uma com os voos de ida e outro com os voos de volta. Só será exibido voos com assentos ainda disponíveis (ou seja, com passagens aéreas que não estão associadas à uma pessoa (i.e. passageiro). A cidade de origem não pode ser a mesma do destino e a data de ida deve ser mais recente do que a data de volta e depois da data atual. Após o usuário selecionar o(s) voos, deve ser feito o cadastro dos passageiros – um para cada passagem comprada - e eles são alocados em assentos aleatórios e, logo após, o cadastro do contato de emergência (um para a reserva). Considere que o cliente já esteja previamente cadastrado.

Grupo 6: Tela CRUD do hotel, a partir de um hotel criado, o CRUD do quarto e, logo após será feita

a tela de reserva de quartos (para criar elementos ReservaQuarto). Na tela de reserva de quartos, o usuário deverá selecionar o período inicial e final da viagem e a cidade que deseja efetuar a reserva e assim, será listado os tipos de quartos disponíveis neste período para a cidade em questão e o preço total (calculado a partir do preço da diária que por temporada). O usuário selecionará um desses tipos de quarto de um determinado hotel e, assim reservará um quarto desse determinado tipo do hotel correspondente. O TipoQuarto e PrecoPorTemporada deve ser cadastrado manualmente.

Grupo 7: CRUD do Quarto, TipoQuarto e PrecoPorTemporada. Faça um CRUD para criação de TipoQuarto. Logo após, os hotéis poderão utilizar os TipoQuarto criados. Os hoteis são criados manualmente. Dado um hotel, é feito uma tela com para o CRUD dos quartos desse determinado hotel. De forma similar, dado o hotel faça o CRUD para o PreçoPorTemporada (o preço é da diária), deverá ter só um preço por período.

Grupo 8: CRUD do Reserva Carro, Carro e Locadora: Faça o CRUD da locadora. Depois, associe seus o CRUD dos carros (associados à ela). Logo após, você fará a tela de reserva de carros. O usuário selecionar o período inicial e final que deseja ficar com o carro e a cidade. O sistema irá listar todos os carros que não possuem reserva nesse período de uma locadora desta cidade. Assim, o usuário poderá selecionar um dos carros e confirmar a sua reserva para este carro no período selecionado. O usuário poderá cancelar a reserva (não será permitido alterá-la).

Classes de gerenciamento dos objetos: estas classes controlarão os objetos e deixaram salvos.
Na primeira fase, cada grupo deverá implementar um (ou mais) destes gerentes. Dependendo das classes a serem implementadas.

GerenteReservaCarros
→ Lista de locadoras
→ Lista de Reserva de carros
→ Método: dado um período e uma cidade retorna os carros disponíveis neste período

GerenteReservaAerea
→ Lista de CiaAereas
→ Lista de Voos
→ Lista de Aeronaves
→ Método: dado uma data uma cidade origem e uma destino, retorna os voos que fazem este trecho

GerenciaPessoasELocalizacao:
→ Lista de pessoas e clientes cadastrados
→ Lista de Cidades
→ Lista recomendações de um cliente

Segunda Fase

Na segunda fase, os grupos deverão se juntar para integrar todo o código feito, eliminando os cadastros manuais quando necessário:

Grupo 1, 3 e 5: Elaboração completa da parte de reserva de passagens aéreas. A cidade ainda poderá ser feita manualmente. O voo só poderá ser removido caso não exista passagens aéreas reservadas.

Grupo 2, 6 e 4: Elaboração completa da parte de reserva de hotéis.

Grupo 7, 8: Elaboração do sistema de Reserva de carro, Reserva de Hotéis e, além disso, uma tela que deverá fazer a reserva de pacotes. A reserva de pacotes, para simplificar, deverá possuir uma reserva de um carro e uma de hotel. O usuário deverá selecionar a cidade e o período correspndente e assim, poder reservar o quarto e o carro correspondente. Tente reutilizar as telas de reserva de hoteis e de carro.

Apêndice

Enumeration:
Geralmente, nossas classes podem instanciar infinitos objetos. Mas, em certas ocasiões, temos um número limitado e conhecido de instâncias para uma classe. Por exemplo, temos apenas 4 estações do ano e, caso desejássemos criar uma classe para estação do ano, ela terá sempre 4 instâncias (objetos). Para isso java tem um recurso que chama **Enumerations**. Ele é usado justamente quando você possui um número limitado de instancias. O funcionamento é similar o de classes porém todos os objetos possíveis de serem criados, são criados pelo próprio **enumeration** – não é possível chamar seu construtor. Por isso, seu construtor sempre será privado.

```
public enum EstacoesAno {
        OUTONO(20,3,19,6),INVERNO(20,6,21,9),
        PRIMAVERA(22,9,20,12), VERAO(21,12,19,3);

        private int diaInicio;
        private int mesInicio;

        private int diaFim;
        private int mesFim;

        private EstacoesAno(int diaInicio,int mesInicio,
                            int diaFim, int mesFim){
                this.diaInicio = diaInicio;
                this.mesInicio = mesFim;

                this.diaFim = diaFim;
                this.mesFim = mesFim;

        }
        (…)
}
```

Logo após criado o Enumeration EstacoesAno, você pode acessar em qualquer parte do seu código as quatro instancias criadas, como se fossem atributos “public static” de EstacoesAno: EstacoesAno.OUTONO, para acessar a instancia outono. Note que é possível alterar dado do objeto, por exemplo, alterar o diaInicio da estação OUTONO. Você só precisará de criar o *setter* correspondente.