TESTE DE SOFTWARE

PROF. EIJI ADACHI M. BARBOSA

Atividade Prática –
Sistema de Cobrança de Estacionamento

Objetivo:

O objetivo desta atividade é exercitar os conceitos de testes funcionais (caixa-preta) e estruturais (caixa-branca) vistos em sala de aula, além de exercitar prática de implementação de testes executáveis com JUnit. Esta atividade vale 5,0 pontos na Unidade 2.

Observação:

Junto a este enunciado, veio um projeto do Eclipse para você iniciar sua implementação. Para tanto, basta você descompactar o .zip e importar o projeto pelo menu *File* > *Import* > *Existing Project*. Caso você use outra IDE, o projeto é um projeto Maven, então basta descompactar o .zip e usar a opção da sua IDE de importar um projeto Maven.

Cobrança de Estacionamento em Aeroportos:

O novo aeroporto da cidade está implantando seu novo sistema de cobranças de taxas de estacionamento. Você é o responsável por implementar e testar tal funcionalidade. A gerência do aeroporto estabeleceu uma série de critérios que devem ser implementados no programa de cobrança. O modelo de cobrança para uso do estacionamento do novo aeroporto prevê três tipos de uso: (i) Estacionamento a Curto Prazo, (ii) Estacionamento a Longo Prazo, e (iii) Estacionamento VIP. O tipo “Estacionamento a Curto Prazo” é pensado nos usuários que usarão estacionamento do aeroporto por apenas algumas horas. O tipo “Estacionamento a Longo Prazo” é pensado nos usuários que usarão o estacionamento do aeroporto por mais do que algumas horas, possivelmente usando por alguns dias. Já o tipo “Estacionamento VIP” é pensado nos usuários que querem conforto e segurança extra ao usar o estacionamento do aeroporto; tal serviço inclui serviço de valet e sala de espera VIP. O usuário deverá definir qual tipo deseja usar já na entrada do estacionamento do aeroporto, pois o tipo deverá ser registrado no sistema junto a hora de entrada. Ao sair do aeroporto, o sistema deverá calcular o valor a ser pago pelo usuário conforme as regras abaixo.

No “Estacionamento a Curto Prazo”, o usuário irá pagar R$ 8,00 pela primeira hora e uma taxa de R$ 2,00 a cada hora extra além da primeira hora; se apenas um minuto passar de uma determinada hora, a taxa de hora extra será cobrada integralmente. Por exemplo: se o usuário entrar às 12:30 e sair as 13:45 do mesmo dia, ele irá pagar R$ 10,00: R$ 8,00 pela hora inicial e R$ 2,00 pelos minutos que excederam a primeira hora. Caso o usuário do “Estacionamento a Curto Prazo” use o estacionamento por mais de 24 horas, ele

deverá pagar uma taxa extra de R$ 50,00, além de pagar a taxa referente às horas extras. A cada 24 horas de permanência esta taxa de R$ 50,00 será cobrada novamente, além da taxa referente à cada hora extra. Após o 7º dia de uso do estacionamento, a taxa de diária extra é reduzida para R$ 30,00.

No “Estacionamento a Longo Prazo”, o usuário irá pagar R$ 70,00 pelas primeiras 24 horas. Após as primeiras 24 horas, o usuário irá pagar uma taxa de R$ 50,00 por cada diária extra; mesmo que apenas um minuto tenha passado do horário que se completa uma diária, a taxa da diária extra será cobrada integralmente. Após o 7º dia de uso do estacionamento, a taxa de diária extra é reduzida para R$ 30,00. Além disso, a cada 30 dias de uso do estacionamento, uma taxa extra de R$ 500,00 é cobrada, mantendo-se ainda a taxa de diária no valor de R$ 30,00.

No “Estacionamento VIP”, o usuário irá pagar uma taxa de R$ 500,00 que lhe dará direito a usar o estacionamento por uma semana. Após a primeira semana, será cobrada uma taxa de R$ 100,00 para cada diária extra. Após o 14º dia de uso do estacionamento, a taxa de diária extra é reduzida para R$ 80,00.

Implementação:
Para implementar o programa que calcula os valores devidos pelos usuários do estacionamento, você irá implementar e testar o módulo que de fato implementa a calculadora do estacionamento. Tal Calculadora deverá ser implementada no método calculateParkingCost da classe Calculator, o qual possui a seguinte assinatura:

Float calculateParkingCost(String checkin, String checkout, ParkingLotType type)

As strings checkin e checkout representam as datas de entrada e saída, respectivamente, e seguem o padrão “yyyy.MM.dd HH:mm”. Ex.: "2017.11.31 10:30". Veja o código disponível na classe Main para ver um exemplo de como manipular strings e datas, além de saber como calcular a diferença entre duas datas.

Considere que os valores válidos para as entradas são:

- Checkin:
  - Ano [1970, 2018]
  - Mês [1, 12]
  - Dia [1, 31]
- Checkout:
  - Ano [1970, 2019]
  - Mês [1, 12]
  - Dia [1, 31]

Caso a string de entrada esteja fora do padrão definido acima, deverá ser lançada a exceção imd0412.parkinglot.exception.DateFormatException. Caso algum valor esteja fora do seu limite, ou uma

data inválida seja passada (ex.: 30 de fevereiro), ou a data de check-out vier antes da data de check-in, lance a exceção imd0412.parkinglot.exception.InvalidDataException. Anos bissextos devem ser testados.

Para este problema, não comece pela implementação do método. Comece pela especificação dos casos de testes. Para isso, comece definindo as classes de equivalência e os limites das entradas. Em seguida, com base nas classes de equivalência, crie uma tabela de decisão para especificar o comportamento esperado do programa e, por fim, crie os casos de testes, cobrindo todas as regras identificadas. Só após toda esta sistematização de especificação do programa e dos casos de testes, comece a implementação do seu programa. Em resumo, siga a prática do *Test-Driven Development* (TDD), incluindo os ciclos de refatoração.

Entregável:

Você deverá implementar classes de testes parametrizáveis: uma para os casos normais e outra para os casos excepcionais, além de uma classe “suíte” para agregar estas classes parametrizáveis. Também faz parte do entregável um documento descrevendo as classes de equivalência, os limites e uma tabela de decisão. Nos casos de testes executáveis implementados com JUnit, deverá ter alguma referência a qual regra da tabela de decisão aquele caso de teste está relacionado (ex.: um comentário indicando algum ID da tabela de equivalência).

Além disso, você deverá implementar o método que calcula a tarifa do estacionamento e desenhar o grafo de fluxo de controle desta implementação. Com base neste grafo, calcule a complexidade ciclomática da sua solução. Por fim, use a ferramenta EclEmma para medir a cobertura dos testes criados. O relatório de cobertura das classes de lógica de negócio devem estar no relatório entregue.

Gere um arquivo .zip nomeado seguindo o padrão SeuNome_-_SeuSobrenome, ou Nome1_-_Sobrenome1-Nome2_-_Sobrenome2. Este arquivo zip deverá conter o projeto ParkingLot com a implementação da calculadora e dos testes. Renomeie o projeto ParkingLot acrescentando o seu nome, ou nome da dupla, como pós fixo (ex.: ParkingLot-Eiji_e_Fulano). Não mude a assinatura do método calculateParkingCost.