Trabalho Prático: Parquímetro

Prof. Daniel Conrado

1 Introdução

Lá em São Carlos-SP, certa região do centro da cidade é chamada de Zona Azul. Neste lugar, a pessoa só pode estacionar por no máximo duas horas. Quando uma pessoa quer estacionar nessa região, ela deve comprar um *ticket* que lhe dá autorização para estacionar por um determinado tempo.

Há vários parquímetros espalhados por essa região. Eles são máquinas que as pessoas utilizam para comprar *tickets* de zona azul. Na Figura 1 há uma foto de uma dessas máquinas.

Um uso típico do parquímetro consiste nos seguintes passos:

1. A pessoa insere moedas no parquímetro. A quantidade inserida é mostrada no visor.
2. Ela aperta o botão azul para *aumentar o tempo* de estacionamento que ela precisa. A Tabela 1 mostra como o tempo aumenta a cada pressionamento.
3. A pessoa aperta o botão verde para confirmar a operação. Um *ticket* (bilhete) é impresso para ela colocar no parabrisa do seu carro. A máquina também devolve o troco, se houver.

1.1 Objetivo

O objetivo deste trabalho prático é criar um projeto Java que simule o funcionamento interno desses parquímetros. Para simplificar, vamos tratar apenas da compra de *tickets* usando moedas. Ficam ignoradas as funcionalidades: pagamentos por cartão, cartões de estacionamento, e o botão Regulariza (eu não sei o que ele faz).

Tabela 1: Variação do tempo do ticket e valores cobrados.

Pressionamento do botão azul	Tempo comprado	Valor cobrado
1 vez	15min	0,50
2 vezes	30min	1,00
3 vezes	1h	1,50
4 vezes	1h30	2,00
5 vezes	2h	2,50

Figura 1: Foto de um parquímetro.

1.2 Método

Está disponível para você uma versão inicial do projeto feita usando o BlueJ. Você vai observar que essa versão não funciona muito bem. Com o projeto aberto no BlueJ, você vai resolver **em sequência** os exercícios aqui listados. Cada exercício modifica ou adiciona uma funcionalidade ao projeto. Findados todos, você deve enviar para mim, via Classroom, o código-fonte das classes, ou seja, os arquivos com extensão **.java**, sem estarem compactados.

2 Exercícios

Exercício 1. Experimente criar um novo objeto da classe **Parquimetro**. Note que o construtor lhe pede como parâmetro o preço da fração. Considerando os preços na Tabela 1, o preço da fração seria 50 centavos. Agora, chame no seu objeto o método **getTempoSolicitado** para ver o tanto de tempo que você está comprando. Chame o método **aumentarTempo** (ele simula o funcionamento do botão azul), e depois chame o método **getTempoSolicitado** novamente.

Agora, insira dinheiro no parquímetro e, depois, chame o método **imprimirTicket** (ele simula o funcionamento do botão verde). Note que este método solicita como parâmetro as horas e os minutos atuais. (Vamos assumir que o parquímetro possui um relógio marcando o tempo e que, quando o usuário aperta o botão verde, a máquina obtém as horas atuais desse relógio e passa como parâmetro para o método **imprimirTicket**.)

Provavelmente você teve que fazer uma conta de cabeça para saber quanto de dinheiro você deveria inserir no parquímetro para pagar o ticket. Seria legal o objeto fornecer uma maneira de consultar o valor a pagar. Insira um novo método chamado **getTotalAPagar** nesta classe. Esse método deve ter **int** como tipo de retorno e não deve receber parâmetros. Ao ser chamado, esse método deve **retornar** o valor total a pagar pelo ticket solicitado.

Exercício 2. Compre um novo ticket de zona azul mas, dessa vez, insira dinheiro *insuficiente* para pagar o ticket e, depois, chame o método **imprimirTicket**. Você vai notar que o ticket *será* impresso! Altere o corpo do método **imprimirTicket** para que o ticket só seja impresso se a quantidade de dinheiro inserida seja igual ou maior ao total a pagar.

Exercício 3. Compre um novo ticket de zona azul mas, dessa vez, insira dinheiro mais que suficiente. Note o que aconteceu. A máquina "engoliu" todas as moedas! Corrija essa lógica em dois passos:

1. Altere o método **imprimirTicket** para que ele cobre do balanço apenas o valor do ticket (ou seja, o valor a pagar). Deixe no balanço o troco.
2. Crie um novo método chamado **devolverTroco** para implementar a devolução do troco. Esse método não recebe parâmetros, deve ter tipo de retorno **int** e deve retornar o troco, assim como *zerar o balanço*.

Exercício 4. Observe que o método **aumentarTempo** atual não segue corretamente a Tabela 1. Faça as devidas correções nesse método. Ainda, impeça que o parquímetro venda tickets acima de 2h.

Exercício 5. Em áreas turísticas, o ticket mínimo dos parquímetros é de 1h. Pense, primeiro, em como você pode alterar a classe **Parquimetro** para que sejamos capazes de criar tanto parquímetros com ticket mínimo de 15min quanto parquímetros com ticket mínimo de 1h. Depois, faça as modificações necessárias.

Exercício 6. Em áreas de grande rotatividade, os parquímetros são configurados para vender um ticket máximo de 1h, e não 2h. Altere a classe **Parquimetro** para que sejamos capazes de criar parquímetros com diferentes tickets máximos.

Exercício 7. Observe que o parquímetro, quando em operação, vai acumulando o dinheiro resultante das vendas dos tickets (para pensar: em qual campo esse dinheiro é acumulado?). Depois de um período em operação, os funcionários da prefeitura passam pelos parquímetros coletando o dinheiro das vendas (até porque, no parquímetro físico, há um limite de moedas que a caixa coletora é capaz de armazenar). Escreva um novo método chamado **esvaziar** para a classe **Parquimetro**. Esse método não recebe parâmetros e deve ter tipo de retorno **int**.

Exercício 8. Você deve ter notado que o ticket impresso pelo parquímetro apresenta as horas e minutos de início do estacionamento e depois o tempo de estacionamento em minutos. Seria mais interessante se o parquímetro mostrasse, ao invés do tempo em minutos, o período final do término do estacionamento. P. ex., se eu compro um ticket de 1h às 9h05, o ticket impresso deve me mostrar que o período de validade do ticket é até às 10h05.

Você provavelmente pensou em resolver este exercício alterando diretamente o método **imprimirTicket**, não foi? Embora possa funcionar, a solução deste desafio não é essa.

Você deve criar uma nova classe que represente um momento no tempo. Objetos dessa classe devem ser criados para representar as horas e minutos de um determinado momento no tempo. Um objeto dessa nova classe deve me fornecer um método para que eu seja capaz de aumentar as suas horas em quantos minutos eu desejar. Esse objeto também deve me fornecer um método que me retorna uma **String** representando as suas horas em formato HH:mm.

Depois de pronta essa classe, altere o método **imprimirTicket** para usar objetos dessa classe para representar o período inicial e final do ticket de estacionamento. Aproveite o método que retorna uma representação em **String** para imprimir corretamente esses períodos no ticket de estacionamento.