Fundamentos de Desenvolvimento de Software
Prof. Bernardo Copstein

Trabalho 1 – Teste de Unidade

Enunciado geral

Uma barca de passageiros tem 1200 lugares organizados em 60 fileiras de 20 lugares cada. O sistema de controle de lugares deve controlar tanto a ocupação dos lugares como a distribuição de peso na barca. Desta forma quando o cliente chega para embarcar ele escolhe um lugar e o sistema deve dizer se o lugar está ocupado ou se ele não pode se sentar ali em função da distribuição de peso. As regras de distribuição de peso são as seguintes:

- Os primeiros 100 passageiros só podem se sentar nas fileiras de 1 a 20.
- Os próximos 100 passageiros só podem se sentar nas fileiras de 40 a 60.
- Os demais passageiros podem sentar-se em qualquer lugar livre.

Os lugares são identificados da seguinte forma: F<nro da fileira>A<nro do assento>. A numeração das fileiras e lugares inicia em 1. Tanto o assento como a fileira têm de ser informados com 2 dígitos.

Exemplos: F02A12, F45A01, F33A18

O custo da passagem da barca varia conforme o horário. Existe um valor base que é válido no horário comercial padrão: 8:00 as 12:00 e das 14:00 às 18:00. No horário das 12:01 às 13:59 e das 18:01 às 19:59 o valor é 10% maior que o preço base. No horário das 20:00 às 23:59 o valor é 20% maior que o preço base e, na madrugada (das 00:00 às 7:59) o valor é 50% maior que o preço base.

A implementação

O software de controle de assentos utiliza uma classe chamada “Barca” que tem como responsabilidades controlar a marcação dos assentos e calcular o preço da passagem. O esqueleto dessa classe pode ser visto a seguir:

```
interface Relogio{
  int getHora();
  int getMinuto();
}
```

```
class Barca{
    public Barca(Relogio relógio, double precoBase){ ... }
    public double defineAssento(String assentoInformado){ ... }
    public void ocupacaoArbitraria(String assentoInformado){ ... }
}
```

A classe “Barca” recebe por injeção de dependência uma classe que implementa a interface “Relogio”. Essa interface prevê métodos que permitem consultar a hora e o minuto correntes. Essas informações são usadas para calcular o preço da passagem com base no “preço base” informado por parâmetro no método construtor.

A classe “Barca” também possui um método chamado “double defineAssento(String assento)” que é usado para marcar o assento que o usuário está comprando e calcular o preço da passagem em função do horário (o assento desejado é informado por parâmetro). O valor de retorno deste método pode ser um valor positivo ou negativo. Se for um valor positivo isso significa que o assento informado por parâmetro pode ser ocupado de acordo com as regras de ocupação dos assentos e o valor retornado corresponde ao preço da passagem. Quando o método retorna o preço da passagem ele também garante que o assento será marcado como ocupado (assume que o valor da passagem será pago). Se o valor for negativo, ele indica um código de erro, e é usado para informar a razão pela qual o assento informado por parâmetro não pode ser ocupado.

Os códigos de erro possíveis são os seguintes:

- (-1.0) – Identificador de assento inválido
- (-2.0) – Assento ocupado
- (-3.0) – Assento bloqueado devido a distribuição de peso

Finalmente a classe “Barca” possui um terceiro método chamado “ocupacaoArbitraria” que é usado para propósitos de teste. Ele marca um assento como ocupado sem fazer verificações. Ele pode ser usado na montagem de cenários para testes de unidade.

Tarefas:

1) Usando as técnicas de particionamento e valor limite, defina um conjunto de casos de teste para o método “defineAssento” da classe “Barca”.
2) Usando o JUnit, implemente drivers de teste a partir dos casos de teste definidos no passo 1.
3) Crie uma classe que implemente a interface “Relogio”.
4) Crie uma implementação para a classe “Barca”.
5) Teste a classe “Barca” usando os testes de unidade disponiveis.
6) Faça a correção dos problemas encontrados.
7) Acrescente novos casos de teste se julgar pertinente.
8) Repita os passos 5, 6 e 7 até que todos os testes sejam aprovados.

Gere um relatório listando cada um dos passos de 1 a 8. Apresente os casos de teste gerados com que técnicas; apresente o código dos drivers de teste; apresente o código original das classes criadas; apresente o resultado da primeira aplicação dos testes; liste os defeitos encontrados; demonstre como os erros foram sanados; indique os testes adicionais propostos (se houverem) e siga demonstrando todo o processo até que todos os erros tenham sido eliminados.

Entregue apenas o relatório final.