Teste e Validação de Software 2024/25
Instituto Superior Técnico

Enunciado do Projecto

Data de Entrega: 23 de Maio de 2025

1 Introdução

O desenvolvimento de grandes sistemas de informação é um processo complexo e requer diversos níveis de abstração. Um primeiro passo é a especificação do problema de uma forma rigorosa. Após a especificação rigorosa do problema, é possível efetuar provas de correção do mesmo ou, caso este esteja incorreto, descobrir falhas que de outra forma seriam difíceis de detectar. Depois, dever-se-á começar a codificar a solução e a testá-la simultaneamente por forma a detetar os erros o mais cedo possível.

O projeto da disciplina Teste e Validação de Software consiste no desenho de casos de teste a partir de uma especificação disponibilizada. Esta especificação diz respeito a algumas entidades de uma dada aplicação. Este projeto tem como objetivo principal de aprendizagem que os alunos ganhem experiência no desenho de casos de testes ao aplicarem as técnicas de desenho de casos de teste leccionadas nas aulas teóricas a um conjunto de classes e métodos cujo comportamento está descrito neste documento.

Este documento está organizado da seguinte forma. A seção 2 descreve as entidades que participam no sistema a testar, assim como alguns detalhes de concretização destas entidades e que serão importantes para os casos de teste a desenhar. A seção 3 identifica os testes a realizar e a forma como a resolução do projeto vai ser avaliada.

2 Descrição do sistema

O sistema a testar é responsável por gerir uma rede de terminais de comunicação. Genericamente, o sistema deverá permitir a gestão e consulta de clientes, terminais e comunicações. De seguida, descrevem-se as principais entidades deste sistema.

2.1 A entidade Cliente

Um cliente tem um nome com no máximo 40 caracteres. O número de terminais de um cliente pode variar entre 1 e 9. Um cliente pode ter *amigos*, que são outros clientes. O número máximo de amigos que um cliente pode ter é determinado pelo número de terminais que possui: $5 * \#terminals - 3$. A lista de amigos de um cliente inicialmente está vazia e um cliente não pode ser amigo de si próprio. Além disso, cada cliente tem um número de pontos que varia de 0 a 200. Quando um cliente é criado, ele começa com 20 pontos. O número de pontos de um cliente afeta o custo das comunicações.

A entidade `Cliente` é representada pela classe `Client`. A listagem 1 apresenta a interface desta classe.

O método `removeTerminal()` remove um terminal, caso seja possível, da lista de terminais do cliente. O método retorna um valor boleano indicando se a remoção foi bem-sucedida. Um terminal pode ser removido apenas se pertencer à lista de terminais do cliente e tiver um saldo não negativo. Se o terminal não pertencer ao cliente, então o método não faz nada e devolve falso. Deve ainda ser considerado que invocações a métodos desta classe, que coloquem o cliente invocado num estado inválido devem ser abortadas, lançando a exceção `InvalidOperationException`, e o estado do cliente em questão não deve ser alterado.

Listing 1: A interface da classe *Client*.

```
package prr.core;

/**
 * This class represents a client.
 **/

public class Client {
  // Creates a client with a single terminal. The provided terminal cannot be null
  public Client(String name, int taxNumber, Terminal term) { /* ... */ }

  public void updateName(name n) { /* ... */ }

  public String getName() { /* ... */ }

  // updates the number of points of the client. It can be a positive or negative number.
  public void updatePoints(int p) { /* ... */ }

  public int getPoints() { /* ... */ }

  public void addFriend(Client c) { /* ... */ }

  public boolean removeFriend(Client c) { /* ... */ }

  public boolean hasFriend(Client c) { /* ... */ }

  public void addTerminal(Terminal terminal) { /* ... */ }

  public boolean removeTerminal(Terminal terminal) { /* ... */ }

  // returns the number of terminals of this client
  public int numberOfTerminals() { /* ... */ }
}
```

2.2 A entidade Terminal

A entidade Terminal permite a realização de comunicações (de voz ou de texto) que envolvem dois terminais, o que inicia a comunicação e o que recebe a comunicação. No caso de uma comunicação de voz, ambos os terminais envolvidos na comunicação mantêm informação sobre a chamada em curso. Cada terminal tem um determinado *saldo* e está associado a um dado cliente. O saldo é definido como a diferença entre o valor das comunicações feitas pelo terminal e os pagamentos feitos pelo cliente. As comunicações realizadas pelo terminal são cobradas de acordo com o tarifário da rede de terminais. O terminal tem ainda uma chave única (cadeia de caracteres numéricos) no contexto da rede de terminais. A listagem 2 apresenta a interface da classe que concretiza a entidade Terminal.

Um terminal recém-criado tem um saldo igual a zero. De seguida, descrevem-se as várias restrições relativamente à invocação dos métodos da classe `Terminal`:

- Quando se cria um terminal, ele fica *desligado* (`OFF`). Neste modo, é possível ligar o terminal (`turnOn()`). Quando um terminal é ligado, ele vai para o modo `NORMAL`. Só é possível realizar um pagamento (`pay()`) no caso de o terminal estar desligado e a quantia a adicionar ao saldo do terminal seja maior ou igual a 5 (a quantia é um valor em cêntimos).

- Um terminal ligado pode estar no modo normal (`NORMAL`), silêncio (`SILENT`) ou ocupado (`BUSY`). O método `toggleOnMode()` permite passar do modo normal para silêncio e vice-versa. Um terminal ligado e não ocupado pode enviar mensagens de texto (`sendSMS()`) para qualquer terminal, inclusivamente para ele próprio. O envio de uma mensagem de texto é instantâneo, mas o facto de um terminal poder enviar a mensagem de texto não quer dizer que o terminal de destino a vá receber. Um terminal só recebe mensagens de texto (`receiveSMS()`) caso esteja ligado. No caso de estar em modo silêncio (independentemente de estar ocupado ou não), há uma restrição adicional para receber uma mensagem de texto: o cliente associado ao terminal que envia a mensagem de texto tem que pertencer ao conjunto de amigos do cliente do terminal que recebe a mensagem de texto.

- Um terminal ligado e não ocupado pode estabelecer uma comunicação de voz com outro terminal através do

Listing 2: A interface da classe *Terminal*.

```
public enum TerminalMode { OFF, NORMAL, SILENT, BUS; }

public class Terminal {

  // creates a terminal with a given identifier and associated to the given client.
  Terminal(String id, Client client) { /* ... */ }

  // Returns the mode of this terminal
  public final TerminalMode getMode() { /* ... */ }

  // Decreases the debt of this terminal by the given amount. The amount must be a number greater than 5 cents
  public void pay(int amount) { /* ... */ }

  // returns the balance of this terminal
  public int balance() { /* ... */ }

  // send a SMS to terminal to with text msg. Returns if the SMS was successfully delivered.
  public boolean sendSMS(Terminal to, String msg) { /* ... */ }

  // receives a SMS from terminal from with text msg
  public void receiveSMS(Terminal from, String msg) { /* ... */ }

  // start a voice call with tetminal to
  public void makeVoiceCall(Terminal to) { /* ... */ }

  // to invoke over the receiving terminal of a voice call (represented by c). The voice
  // call is established if the terminal accepts the call, otherwise it throws an exception.
  void acceptVoiceCall(Communication c) { /* ... */ }

  // turns on this terminal
  public void turnOn() { /* ... */ }

  // turns off this terminal
  public void turnOff() { /* ... */ }

  // toggles the On mode: normal to silent or silent to normal
  public void toggleOnMode() { /* ... */ }

  // Ends the ongoing communication.
  public void endOngoingCommunication() { /* ... */ }
}
```

método `makeVoiceCall()` desde que o terminal destinatário possa receber chamadas de voz. Um terminal ligado e não ocupado pode aceitar uma comunicação de voz (`acceptVoiceCall`) se o terminal estiver no modo *NORMAL*. Caso seja possível estabelecer a comunicação de voz, então os dois terminais envolvidos passam a estar ocupados durante a duração da chamada, voltando para o modo anterior quando a chamada terminar (`ongoingCommunicationEnded()`).

- É possível saber o saldo de um terminal (`balance()`) desde que o terminal não esteja ocupado. Um terminal ligado e não ocupado pode ser desligado através do método `turnOff()`.

O método `getMode()` pode ser invocado em qualquer situação e devolve o modo do terminal. Caso a invocação dos métodos indicados não ocorra numa das situações descritas, então o método não deverá ter qualquer efeito e deverá lançar a exceção *InvalidInvocationException*.

2.3 A entidade Communication

As comunicações entre dois terminais são representadas pela classe `Communication`. Uma comunicação já finalizada tem um dado tamanho. O tamanho de uma comunicação de texto é igual ao número de caracteres da mensagem de texto associada à comunicação em causa (medido em centenas e arredondado para cima). O tamanho de uma comunicação de voz é dado pela duração (em segundos) da comunicação. A listagem 3 apresenta a interface desta classe:

O custo de uma comunicação depende do tipo de comunicação, da sua duração, do número de pontos e do número

Listing 3: A interface da classe *Communication*.

```
package prr.core;

public enum CommunicationType { SMS, VOICE; }

/**
 * This class represents a communication (text or voice) made between
 * two terminals.
 **/
public class Communication {

  private Communication(CommunicationType type, Terminal to, Terminal from) { /* ... */ }

  public static Communication textCommunication(Terminal to, Terminal from, int length) { /* ... */ }

  public static Communication textCommunication(Terminal to, Terminal from) { /* ... */ }

  public void duration(int duration) { /* ... */ }

  public Terminal to() { /* ... */ }

  public Terminal from() { /* ... */ }

  double computeCost() { /* ... */ }

  public double getCost() { /* ... */ }
}
```

de amigos do cliente do terminal que iniciou a comunicação. O custo de uma comunicação é então calculado da seguinte forma:

- Se o tamanho da comunicação for igual a 0, então o custo é 0 cêntimos;
- Se o tamanho da comunicação for inferior a 10 e o número de pontos do cliente for superior a 100, então o custo é 1 cêntimo. Se o número de pontos for menor ou igual a 100, então o custo é 2 cêntimos.
- Se o tamanho for maior ou igual a 10 e inferior a 120, então é necessário considerar o número de pontos do cliente que iniciou a comunicação e o tipo de comunicação. Se o número de pontos for inferior a 75, então o custo é 6 cêntimos para uma mensagem de texto e 12 cêntimos para uma comunicação de voz. Se o número de pontos for maior ou igual a 75, então o custo da comunicação é 4 cêntimos para uma comunicação de texto. No caso de uma comunicação de voz, é necessário ainda considerar o número de amigos do cliente. Se o número de amigos for inferior a 4, o custo é 8 cêntimos. Se o número de amigos for maior ou igual a 4, então o custo é 5 cêntimos.
- Se o tamanho da comunicação for maior ou igual a 120 e o número de pontos for inferior a 150, então o custo da comunicação é 15 cêntimos. Se o número de pontos for maior ou igual a 150, então o custo é 12 cêntimos.

3 Avaliação do projeto

Todos os casos de teste a desenhar devem ser determinados aplicando os padrões de desenho de testes mais apropriados. O projeto consiste na especificação dos casos de teste para testar alguns métodos e algumas classes ao nível de classe. Será ainda necessário concretizar alguns dos casos de teste especificados. Os casos de teste a especificar são os seguintes:

- Casos de teste ao nível de classe da classe `Client`. Valem entre 0 e 3,5 valores.
- Casos de teste ao nível de classe da classe `Terminal`. Valem entre 0 e 6,5 valores.
- Casos de teste correspondentes ao método `computeCost()` da classe `Communication`. Valem entre 0 e 3,5 valores.

- Casos de teste correspondentes ao método `removeTerminal()` da classe `Client`. Valem entre 0 e 3,5 valores.

A especificação dos casos de teste deve ser realizada aplicando a estratégia de desenho dos testes mais apropriada para cada situação. **Adicionalmente**, é necessário concretizar 8 casos de teste (4 casos de teste de *sucesso* e 4 casos de teste de *insucesso*) da bateria de testes desenhada para exercitar a classe `Client` ao nível de classe. Esta concretização deve ser feita utilizando a framework de testes *TestNG* e vale entre 0 e 3 valores.

Caso aplique a estratégia de desenho de teste *Category Partition*, se o número de combinações a apresentar for maior do que 30, apenas é necessário apresentar as primeiras 30 combinações e indicar o número total de combinações. Para cada método ou classe a testar é necessário indicar o seguinte:

- O nome da estratégia de desenho de testes aplicada.
- Caso seja aplicável, indicar o resultado dos vários passos da estratégia aplicada, utilizando o formato apresentado nas aulas teóricas.
- A especificação dos casos de teste resultantes da aplicação da estratégia de desenho de tested escolhida. Normalmente, a especificação corresponde ao resultado obtido no último passo da estratégia.

Caso algum dos pontos mencionados acima apenas seja satisfeito parcialmente, a nota será dada na proporção realizada.

3.1 Detecção de cópias

A submissão de um projeto pressupõe o **compromisso de honra** que o trabalho incluso foi realizado pelos alunos referenciados nos ficheiros/documentos submetidos para avaliação. A quebra deste compromisso, ou seja, a tentativa de um grupo se apropriar de trabalho realizado por colegas, tem como consequência a reprovação nesta disciplina de todos os alunos envolvidos (incluindo os que possibilitaram a ocorrência).

3.2 Notas Finais

Poderá existir uma discussão que avalie a capacidade dos alunos em realizar testes semelhantes aos realizados no projeto. Esta decisão cabe exclusivamente ao corpo docente da cadeira. Alunos cuja nota no exame seja inferior em mais de 5 valores à nota do projeto poderão ter de realizar uma discussão. Caso haja lugar a uma discussão, a nota final do projeto poderá ser individualizada e será a nota obtida na discussão.

Toda a informação referente ao projeto estará disponível na página da disciplina na seção *Projeto*. O protocolo de entrega do projeto está disponível nesta seção da página da disciplina.

Bom Trabalho!