Testes e Validação de Software 2015/16
Instituto Superior Técnico
Enunciado do 1º Projecto

Data de Entrega: 8 de Abril de 2016

1 Introdução

O desenvolvimento de grandes sistemas de informação é um processo complexo e requer diversos níveis de abstracção. Um primeiro passo é a especificação do problema de uma forma rigorosa. Após a especificação rigorosa do problema, é possível efectuar provas de correcção do mesmo ou, caso este esteja incorrecto, 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 detectar os erros o mais cedo possível.

O projecto de Testes e Validação de Software consiste no desenho de casos de teste a partir de uma especificação dada. Esta especificação diz respeito a um sistema, designado como *ZOO*, que representa um simulador de um jardim zoológico.

Este projecto tem como objectivo principal de aprendizagem que os alunos ganhem experiência no desenho de casos de testes aplicando os padrões de desenho de teste leccionados nas aulas teóricas.

A secção 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. Finalmente, a secção 3 identifica os testes a realizar e a forma como a resolução do projecto vai ser avaliada.

2 Descrição do sistema

O sistema a testar simula a gestão de um jardim zoológico. Este sistema é responsável por manter a informação relacionada com um dado jardim zoológico: animais, habitats e veterinários. De seguida descrevem-se as principais entidades do *ZOO*.

Listing 1: A interface da classe *Zoo*.

```
public class Zoo {
    public Zoo(String name, int area, int maxAnimals) { /* .... */ }
    /* Animal related methods */
    // Adds an animal to Zoo
    public void add(Animal a) throws InvalidOperationException { /* .... */ }
    // Removes the animal from the Zoo if it is not in any habitat of the Zoo.
    // Returns true in this case. Otherwise, returns false and the animal is not removed.
    public boolean remove(Animal a) { /* .... */ }
    // Returns all animals of the zoo
    public List<Animal> getAnimals() { /* .... */ }
    // Returns true if the animal belongs to the zoo, false otherwise.
    public boolean has(Animal animal) { /* .... */ }
    /* Habitat related methods */
    // Adds an habitat to the zoo
    public void add(Habitat h) { /* .... */ }
    // Removes an empty habitat from the zoo.
    public void remove(Habitat h) throws InvalidOperationException { /* .... */ }
    // Put the animal of Zoo in the habitat of the Zoo. If the animal is already in
    // another habitat it is removed from that habitat.
    public void placeIn(animal animal, Habitat habitat) throws InvalidOperationException
        { /* .... */ }
    // Returns all habitats of the zoo
    public List<Habitat> getHabitats() { /* .... */ }
    // some other stuff
    public int getArea() { /* .... */ }
    public String getName() { /* .... */ }
}
```

2.1 A entidade Zoo

A entidade *Zoo* mantém os animais e habitats que pertencem ao *Zoo*. Um *Zoo* tem um nome e ocupa uma dada área. Num dado instante, cada animal do Zoo pode estar no máximo num habitat do *Zoo* (é possível não estar em nenhum).

Em qualquer momento, podem ser criados novos habitats ou removidos habitats antigos (desde que estejam vazios). O somatório das áreas dos habitats do *Zoo* não pode ser maior do que a área do *Zoo*. A listagem 1 apresenta a interface da classe *Zoo*.

2.2 A entidade Animal

Cada animal tem um nome e uma espécie e é representado pela classe *Animal*. Cada espécie é identificada pelo seu nome. Os animais do Zoo são mantidos por tratadores, que são responsáveis por distribuir a comida e pela limpeza dos habitats, e por veterinários, que são responsáveis por zelar pela saúde dos animais. Os veterinários são ainda responsáveis por vacinar os animais. A listagem 2 descreve a interface da classe *Animal*.

Relativamente à entidade *Animal* existem as seguintes restrições relativamente à invocação dos seus métodos:

- Um animal quando é criado fica de boa saúde e acordado. Um animal de boa

Listing 2: A interface da classe *Animal*.

```
public class Animal {

    public static enum AnimalState {SLEEPING, DEAD, AWAKE}

    public Animal(String name, String species} { /* .... */ }

    public boolean vaccinate(Vaccine vaccine) throws InvalidOperationException
    { /* .... */ }
    public void sicken() throws InvalidOperationException { /* .... */ }
    public void die() throws InvalidOperationException { /* .... */ }
    public void sleep() throws InvalidOperationException { /* .... */ }
    public void awake() throws InvalidOperationException { /* .... */ }
    public void cure(Veterinarian vet) throws InvalidOperationException { /* .... */ }

    // Access methods
    public AnimalState getState() { /* .... */ }
    public boolean isSick() { /* .... */ }
    ...
}
```

saúde e acordado pode ser vacinado (método *vaccinate*), adoecer (método *sicken*), adormecer (método *sleep*) e morrer (método *die*). Relativamente ao método *vaccinate*, se a vacina não for apropriada à espécie do animal a vacinar, então o animal ficará doente. Não há problemas em vacinar um animal com uma vacina que ele já tenha tomado antes.

- Um animal acordado e doente pode ser vacinado, morrer, adormecer e ser curado (método *cure*). Se o veterinário encarregue de tratar o animal souber curar a espécie do animal, então este ficará de boa saúde, caso contrário morrerá. Relativamente à vacinação, se a vacina não for apropriada à espécie do animal, então este morrerá.

- Um animal a dormir pode acordar (método *awake*), preservando o estado em que estava antes de adormecer (doente ou de boa saúde).

- Finalmente, um animal morto não come, não dorme, não acorda, não pode ser vacinado nem curado, não adoece e não morre.

Caso a invocação dos métodos indicados não ocorra numa das situações descritas, então o método não deverá ter qualquer efeito e deverá lançar a excepção *InvalidInvocationException*. Os métodos *isSick* e *getState* podem ser invocados em qualquer situação.

2.3 A entidade Habitat

Um habitat mantém animais do *Zoo*. Por razões de saúde, o espaço médio disponível por animal num habitat tem que ser no mínimo de 4 m^2. Cada habitat tem uma dada capacidade máxima que não pode ser ultrapassada. Um habitat

Listing 3: A interface da classe *Habitat*.

```
public class Habitat {
    public Habitat(int area, List<Animal> animals, int maxAnimals, Zoo zoo) throws
        InvalidOperationException
    { /* .... */ }
    void add(Animal animal) throws InvalidOperationException { /* .... */ }
    void remove(Animal animal) throws InvalidOperationException { /* .... */ }
    public void setArea(int area) throws InvalidOperationException { /* .... */ }
    public int getArea() { /* .... */ }
    public boolean has(Animal animal) { /* .... */ }
    public List<Animal> getAnimals() { /* .... */ }
    public Zoo getZoo() { /* .... */ }
    public int getSatisfaction(Animal a) throws NotInHabitatException { /* .... */ }
    ...
}
```

só pode conter animais da mesma espécie e que pertencem ao mesmo *Zoo* que o habitat. Qualquer habitat tem que conter no mínimo um animal. A listagem 3 apresenta a interface da classe Habitat. A invocação dos métodos *Habitat*, *add*, *remove* e *setArea* lançam a excepção *InvalidOperationException* caso a operação a realizar torne o habitat inválido. No caso da excepção ser lançada, o estado original do habitat não deve ser alterado.

Cada animal do *Zoo* num dado habitat tem um dado grau de satisfação, o qual depende do estado do animal e do espaço disponível no habitat para cada animal, de acordo com as seguintes regras (por ordem decrescente de prioridade):

- Se o animal não pertencer ao habitat então é lançada a excepção *NotInHabitatException*;
- Se o animal estiver morto, o seu grau de satisfação é -100;
- Se for o único animal do habitat, o seu grau de satisfação é -15;
- Se o animal estiver de boa saúde e o espaço disponível por animal no habitat for:
  - menor do que 10 m^2, o grau de satisfação é 15;
  - entre 10 m^2 e 20 m^2 (inclusive), o grau de satisfação é 30;
  - maior do que 20 m^2, o grau de satisfação é 35;
- Se o animal estiver doente e o espaço disponível por animal no habitat for:
  - menor ou igual do que 20 m^2, o grau de satisfação é 5;
  - maior do que 20 m^2, o grau de satisfação é 15;

Listing 4: A interface da entidade *Veterinário*.

```
public class Veterinarian {
 public Veterinarian(String name, List<String> species) { /* .... */ }
 public void cure(Animal a) throws CannotCureAnimalException { /* .... */ }
 public void vaccinate(Animal a, Vaccine vaccine) { /* .... */ }
 public void rest() { /* .... */ }
 public String getName() { /* .... */ }
 public boolean knows(String species) { /* .... */ }
}
```

2.4 A entidade Veterinário

Os veterinários vacinam e curam animais. Cada veterinário sabe vacinar animais de todas as espécies mas só sabe curar animais de determinadas espécies. Cada veterinário conhece o conjunto de espécies que sabe curar. Se um veterinário tentar curar um animal (doente) de uma espécie que ele não sabe curar, então o animal morre. A listagem 4 apresenta a interface da entidade *Veterinário*.

Um veterinário sem descansar pode tratar até 100 animais. Após tratar este número, o veterinário fica cansado e já não pode tratar mais animais enquanto estiver cansado. Um veterinário fica descansado após a invocação do método *rest*. A excepção *CannotCureAnimalException* é lançada caso o animal a curar não possa ser curado pelo veterinário de acordo com as regras de negócio indicadas nas secções 2.2 e 2.4.

3 Avaliação do projecto

Todos os casos de teste a realizar devem ser desenhados aplicando os padrões de desenho de testes mais apropriados. Os casos de testes a realizar por cada grupo correspondem a testar alguns métodos e classes das entidades descritas anteriormente:

- Casos de teste ao nível de classe da classe *Habitat*. Valem entre 0 e 3,5 valores.
- Casos de teste ao nível de classe da classe *Animal*. Valem entre 0 e 6,5 valores.
- Casos de teste correspondentes ao método *getSatisfaction* da classe *Habitat*. Valem entre 0 e 3,5 valores.
- Casos de teste correspondentes ao método *cure* da classe *Veterinarian*. Valem entre 0 e 3,5 valores.
- Adicionalmente, é necessário concretizar 6 casos de teste da bateria de testes que testa a classe *Habitat*. Esta concretização deve ser feita utilizando a framework de testes *TestNG* e vale entre 0 e 3 valores.

Para cada método ou classe a testar é necessário indicar o seguinte:

- O nome do padrão de teste aplicado.
- Caso seja aplicável, indicar o resultado dos vários passos da aplicação do padrão, utilizando o formato apresentado nas aulas teóricas.
- A descrição dos casos de teste resultantes da aplicação do padrão de teste escolhido.

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

3.1 Discussão do projecto

Poderá existir uma discussão que avalie a capacidade dos alunos em realizar testes semelhantes aos realizados no projecto. Esta decisão cabe exclusivamente ao corpo docente da cadeira. Alunos cuja nota no primeiro teste seja inferior em mais de 5 valores à nota do projecto terão de realizar uma discussão e neste caso a nota do projecto será individualizada e dependerá do desempenho na discussão.

3.2 Detecção de cópias

A submissão de um projecto 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).

3.3 Notas Finais

O prazo de entrega do projecto é dia **8 de Abril de 2016 às 16:59:00**. O protocolo de entrega do projecto será disponibilizado oportunamente na página da cadeira bem como qual a documentação a entregar.

Toda a informação referente a este projecto estará disponível na página da cadeira na Secção *Projecto*. O esclarecimento de dúvidas deve ser feito preferencialmente nas sessões de dúvidas.

Bom Trabalho!