Ficha Prática 8

António Nestor Ribeiro, António Fernandes, F. Mário Martins
{anr,af,fmm}@di.uminho.pt
PPIV (LESI)

2003/04

Objectivos

1. Herança. Classes e métodos abstractos.
2. Excepções em Java.
3. Exercícios.

Excepções

O mecanismo de excepções em Java, permite que o programador construa as suas próprias classes de excepção, de forma muito simples, por herança da superclasse das excepções, a classe `Exception`.

O código necessário para criar uma classe de excepções toma a seguinte forma:

```
class MinhaExcepcao extends Exception {
   MinhaExcepcao() {super();}
   MinhaExcepcao(String s) {super(s);}
}
```

Lançar uma excepção

Quando existe uma situação de erro, por exemplo retirar um elemento da colecção quando ele não existe, o programador pode explicitamente sinalizar o envio de um erro que deve ser posteriormente tratado.

Para tal apenas é necessário criar uma instância da excepção pretendida, da forma:

```
...
throw new <nomeClasseExcepcao>(); //construtor vazio

...
...
throw new <nomeClasseExcepcao>("mensagem que se queira enviar");
                        //construtor parametrizado por uma String
```

A única forma que um programador tem de saber que um determinado método pode lançar uma excepção, é isso estar explicitamente reflectido na assinatura do método.

Para tal a assinatura do método deve ser feita de forma a indicar que o método pode "libertar"alguma excepção. Isso deve ser feito de acordo com a seguinte sintaxe:

```
public <tipoDados> <nomeMétodo>(<listaParametros>)
                                throws <listaClassesExcepção> {

...

}
```

Um método que invoque outro que pode lançar uma excepção, deve precaver-se e ter disponível código de tratamento para essa excepção. Se não o fizer, então esse método deve explicitamente (na sua claúsula de `throws`) declarar que também pode libertar uma excepção desse tipo.

Apanhar uma excepção

Como referido atrás ao invocar um método que pode lançar uma excepção, o programador deve prever a existência de código de tratamento do erro, por forma a que o programa não acabe de forma brusca, devido ao facto de a excepção não ter sido devidamente identificada e tratada.

A construção Java que permite apanhar as excepções e tratá-las tem a seguinte forma:

```
try {
  // código que liberta excepções
  ...
  ...

}

catch (<tipoExcepção1>) {
  //tratamento da excepção
}

catch (<tipoExcepção2>) {
  //tratamento da excepção
}

catch (...) { ... }

finally {
  // código que pode ser executado após as claúsulas
  // de catch

}
```

Tenha em linha de conta que esta estrutura é muito semelhante a um `switch`, logo após entrar numa zona de tratamento a uma classe de excepção, sai da estrutura try/catch e só pode exectuar o código que estiver em finally.

A zona de finally é opcional e pode não existir.

Exercícios

1. Na classe `CarteiraAccoes` crie as classes de excepção necessárias e codifique as situações em que deve lançar excepções. Sugere-se que valide as pré-condições recorrendo a esta nova técnica de programação (ex: remover uma acção não existente, dar a carteira de um titula não existente, etc.)

2. Acabe a classe `Corretora` começada na ficha anterior e crie as excepções necessárias para tornar robusto e fiável o código produzido.

   Recorda-se que correctora deverá ter os seguintes métodos:

(a) i. `public double montanteGerido()` - dá o montante investido pelos titulares;

ii. `public Set carteirasComLucro()` - dá um conjunto com os titulares da carteira que no momento estão a ter lucro;

iii. `public String titularComMaisLucro()` - dá o nome do titular cuja carteira apresenta a maior valorização;

iv. `public Set titularesDeAccao(String empresa)` - dá um conjunto com os titulares que possuem uma determinada acção.

(b) Implemente também os métodos `toString`, `equals` e `clone`.