Universidade Federal do Ceará - Campus de Quixadá

Programação para Design

Exceções

Prof. Anderson Lemos

Introdução

- Exceções são:
  - Erros em tempo de execução.
  - Objetos criados a partir de classes especiais que são “lançados” quando ocorrem condições excepcionais.
- Métodos podem capturar ou deixar passar exceções que ocorrerem em seu corpo.
  - Em algumas linguagens, existem exceções em que o tratamento é obrigatório.
  - Em TypeScript não é obrigatório.
- Operador *try-catch* é usado para tentar capturar exceções enquanto elas passam por métodos.

Tipos de erros em tempo de execução

- Erros de lógica de programação.
  - Ex: limites do vetor ultrapassados, divisão por zero.
  - Devem ser corrigidos pelo programador.
- Erros devido a condições do ambiente de execução.
  - Ex: arquivo não encontrado, rede fora do ar, etc.
  - Fogem do controle do programador mas podem ser contornados em tempo de execução.
- Erros graves onde não adianta tentar recuperação.
  - Ex: falta de memória, erro interno.
  - Fogem do controle do programador e não podem ser contornados.

Como usar uma exceção?

- Uma exceção é um tipo de objeto que sinaliza que uma condição excepcional ocorreu.
  - A identificação (nome da classe) é sua parte mais importante
- Precisa ser criada com ***new*** e depois lançada com ***throw***

```
let e : RangeError = new RangeError("Erro!");
throw e; //lançando a exceção
```

- A referência é desnecessária. A sintaxe abaixo é mais usual:

```
throw new RangeError("Erro!");
```

O que acontece?

- Uma exceção lançada interrompe o fluxo normal do programa.
  - O fluxo do programa segue a exceção.
  - Se o método onde ela ocorrer não a capturar, ela será propagada para o método que chamar esse método e assim por diante.
  - Se ninguém capturar a exceção, ela irá causar o término da aplicação.
  - Se em algum lugar ela for capturada, o controle pode ser recuperado.

Captura e lançamento de exceções

```
class RelatorioFinanceiro {

    public metodoMau() : void {
        let dadosCorretos : boolean = verificaDados();
        if(!dadosCorretos){
            throw new RelatorioError("Dados incorretos");
        }
    }

    public metodoBom() : void {
        try{
            console.log("Entrando no método bom");// É executado sempre
            this.metodoMau();
            console.log("Depois do método mau"); // É executado se a exceção não for lançada
        } catch(ex){
            console.log(ex);
        }
        console.log("Saindo do método bom"); // É executado se a exceção não ocorrer ou for capturada
    }
}
```

try, *catch* e *finally*

- O bloco *try* "tenta" executar um bloco de código que pode causar exceção.
- Deve ser seguido por:
  - Obrigatoriamente por um bloco *catch*.
  - Opcionalmente por um um bloco *finally*.
- Blocos *catch* recebem uma exceção como argumento.
  - Se ocorrer uma exceção no *try*, ela irá executar o bloco *catch*.
- O bloco *finally* contém instruções que devem se executadas independentemente da ocorrência ou não de exceções.

try, *catch* e *finally*

```
try{
    // Instruções que executam até ocorrer uma exceção
} catch(ex){
    // Executa somente se ocorrer alguma exceção
} finally {
    // Executa sempre
}
```

Exceções nativas do TypeScript

- TypeScript tem por padrão a classe Error, que é base para todas as exceções da linguagem. Essa classe é o pai de todas as exceções.
- Algumas outras exceções de TypeScript são:
  - EvalError
  - InternalError
  - RangeError
  - ReferenceError
  - SyntaxError
  - TypeError
  - URIError

Como criar nossas exceções

- Para criar uma classe que represente sua exceção, basta:
  - Estender a classe *Error* (que já é definida pelo TypeScript).
  - **Implementar a interface *Error* (que já é definida pelo TypeScript).**
    - Essa última opção é mais simples, em termos de implementação.
- Não precisa de mais nada. O mais importante é implementar *Error* e fornecer uma identificação diferente.
  - Bloco *catch* usa nome da classe para identificar exceções.

Como criar nossas exceções

- É obrigatório que a sua classe tenha dois atributos **públicos**: *name* e *message*.
- Você deve setar o valor de *name* com o mesmo nome da sua classe, sendo assim, não precisa ser passado no construtor.
- O valor de *message* é passado pelo construtor.
- Você também pode acrescentar métodos e atributos como em qualquer classe.

Como criar nossas exceções

```
class MyError implements Error{
    public name : string;
    public message : string;
    public constructor(message : string){
        this.name = "MyError";
        this.message = message;
    }

    public toString() : string {
        return this.name+": "+this.message;
    }
}
```

Como pegar qualquer exceção

- A exceção será capturada pelo bloco *catch*.
- No bloco *catch*, você pode verificar o tipo da exceção com o operador ***instanceof***.
  - Assim é possível tratar exceções de formas diferentes, dependendo de qual exceção foi lançada.

```
try{
    // instruções
} catch(ex){
    if(ex instanceof MyError){
        // trata MyError
    } else if(ex instanceof ReferenceError){
        // trata ReferenceError
    } else {
        // faz alguma outra coisa
    }
}
```

Como relançar uma exceção

- Às vezes, após a captura de uma exceção, é desejável relançá-la para que outros métodos lidem com ela.
  - Isto pode ser feito da seguinte forma:

```
try{
    // instruções
} catch(ex){
    // faz alguma coisa para lidar com a exceção
    throw ex;
}
```

Tipos de exceções

- Em algumas linguagens podem existir exceções verificadas e não verificadas.
  - Ex: JAVA tem as classes *RuntimeException*, *Exception* e *Error*.
  - Em TypeScript todas as exceções são como *RuntimeException*.

Como cavar a própria cova

- Não tratar exceções ou deixá-las para algum método que chamou tratá-las evita trabalho, mas torna o código menos robusto.
- Mas o pior que um programador pode fazer é capturar exceções e fazer nada, permitindo que erros graves passem despercebidos e causem problemas dificílimos de localizar no futuro.
- Não é uma boa prática de programação tentar capturar a exceção *Error* ou lançar uma exceção desse tipo.
  - Não é possível saber qual tipo de exceção foi lançada
  - É desejável ser o mais específico possível no tratamento de erros.

Exercício

- Calculadora
  - Calcula as quatro operações básicas (+, -, *, /)
  - A calculadora deverá pegar os dados do usuário, e fazer várias operações, até o usuário digitar o comando “sair”.
  - Crie a classe *NotAOperationError* que implementa *Error*.
    - Se o usuário digitar uma operação inválida essa exceção deve ser lançada.
  - Crie a classe *NotANumberError* que implementa *Error*.
    - Se o usuário digitar um valor que não possa ser convertido para *number* essa exceção deve ser lançada.
    - Dica: use as funções *Number* e *isNaN*.
  - Crie a classe *DivisionZeroError* que implementa *Error*.
    - Caso haja uma tentativa de divisão por zero, o programa deve lançar essa exceção.

Perguntas?