



Programação Orientada a Objetos

Exceções - Parte 2

Ely – ely.miranda@ifpi.edu.br

1

Criando novas exceções

- A classe Error é a classe “mãe” das exceções;
- É comum termos exceções específicas para a nossa aplicação a fim de filtrarmos os erros;
- Tais exceções herdam da classe Error.

ely.miranda@ifpi.edu.br

2

2

Criando novas exceções

- Herdando da classe Error:

```
class AplicacaoError extends Error {  
    constructor(message: string) {  
        super(message);  
    }  
}
```

ely.miranda@ifpi.edu.br

3

3

Criando novas exceções

- Herdando da AplicacaoError:

```
class SaldoInsuficienteError extends  
    AplicacaoError {  
    constructor(message : string) {  
        super(message);  
    }  
}
```

ely.miranda@ifpi.edu.br

4

4

Lançando uma exceção personalizada

```
class Conta {
    //...
    sacar(valor: number): void {
        if (this._saldo < valor) {
            throw new SaldoInsuficienteError('Saldo insuficiente.');
        }

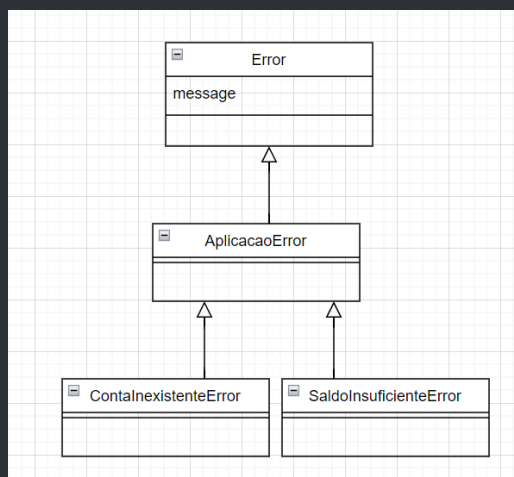
        this._saldo = this._saldo - valor;
    }
}
```

ely.miranda@ifpi.edu.br

5

5

Hierarquia de classes



ely.miranda@ifpi.edu.br

6

6

Tratamento de exceções

- Uma exceção quando lançada finaliza a execução a partir do erro;
- Nossas aplicações devem ser robustas e executar suas funções mesmo em condições anormais;
- Devemos então tratar as exceções de forma que torne o código robusto e que volte a operar normalmente.

ely.miranda@ifpi.edu.br

7

7

Tratamento de exceções

- O tratamento de exceções se dá com o uso de blocos chamados try/catch;
- Você envolve o código em um bloco try { };
- Caso uma exceção seja lançada, capturamos e tratamos o erro no bloco catch { }.

ely.miranda@ifpi.edu.br

8

8

Try/catch

```
let c: Conta = new Conta("1",100);
try {
    c.sacar(100);
    c.sacar(1000);
    c.depositar(10); //não é executada
} catch (e: any) {
    console.log(e.message);
}
```

ely.miranda@ifpi.edu.br

9

9

Bloco catch

- Recebe um tipo de exceção como argumento;
- Se ocorrer uma exceção no try, o fluxo irá pular para o catch;
- Caso seja necessária, outra exceção pode ser lançada dentro do catch.

ely.miranda@ifpi.edu.br

10

10

E se o erro não for AplicacaoError?

```
let c: Conta = new Conta("1",100);
try {
    throw new Error("Um erro de conexão qualquer...");
    c.sacar(1000); //não é executada
} catch (e: any) {
    if (e instanceof AplicacaoError) {
        console.log(e.message);
    }
    //... outros ifs ou elses
    if (e instanceof Error) {
        console.log("Erro no sistema. Contate o administrador.");
    }
}
```

ely.miranda@ifpi.edu.br

11

11

E se o erro não for AplicacaoError?

- Sempre devemos prever de forma genérica erros da aplicação;
- Começamos o tratamento das exceções pela classe por nós definidas, no caso AplicacaoException;
- Os últimos tratamentos devem ser da classe Error, pois esses não foram a princípio detectados como possíveis.

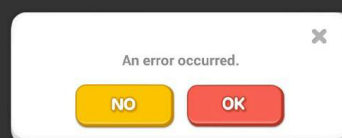
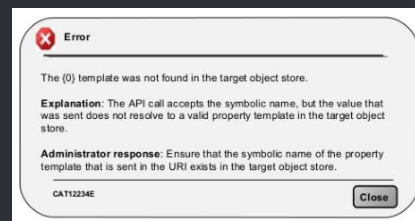
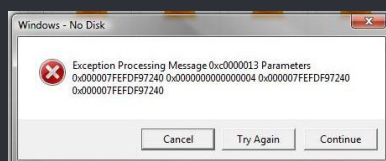
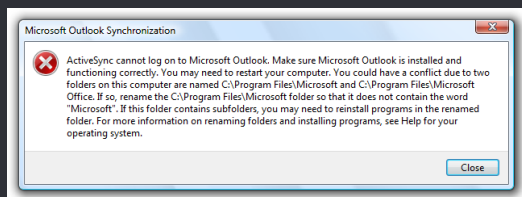
ely.miranda@ifpi.edu.br

12

12

E se o erro não for AplicacaoError?

- Dessa forma, podemos omitir dados técnicos da falha.



ely.miranda@ifpi.edu.br

13

13

Bloco finally

- Utilizado como último bloco após o bloco catch;
- É executado sempre: independente de exceção ou não;
- Serve para liberar recursos, fechar conexões e arquivos e dar mensagens que ocorrem em todos os casos.

ely.miranda@ifpi.edu.br

14

14

Bloco finally

```
let c: Conta = new Conta("1",100);
try {
    //...
} catch (e: any) {
    //...
} finally {
    console.log("Operação finalizada");
}
```

ely.miranda@ifpi.edu.br

15

15

Resumindo os blocos try/catch/finally

- Trechos de código usados para tratar exceções;
- O bloco try "tenta" executar um bloco de código que pode lançar ou levantar exceção;

```
try {
    //código "gerenciado", que pode gerar erros
}
```

ely.miranda@ifpi.edu.br

16

16

Resumindo os blocos try/catch/finally

- Um bloco try deve ser seguido por:

- Um catch;
- E/ou um bloco finally.

```
try {
    //...
}
catch (e: any) {
    // tratamento da exceção
}
```

ely.miranda@ifpi.edu.br

17

17

Resumindo os blocos try/catch/finally

- Um bloco try deve ser seguido por:

- Um catch;
- E/ou um bloco finally.

```
try {
    //...
} catch (e: any) {
    // tratamento da exceção
} finally {
    // código sempre executado
}
```

ely.miranda@ifpi.edu.br

18

18

Aplicação robusta

```
let b: Banco = new Banco();
let opcao: string = "";
do {
    //exibir menu com opções
    try {
        //ler uma opção pelo teclado
        switch (opcao) {
            case "1":
                //opção cadastrar...
                break
            case "2":
                break
            //opção excluir
            //demais casos...
        }
    } catch (e: any) {
        if (e instanceof AplicacaoError) {
            console.log(e.message);
        }
        //... outros ifs ou elses
        if (e instanceof Error) {
            console.log("Erro no sistema. Contate o administrador.");
        }
    } finally {
        console.log("Operação finalizada. Digite 9 para sair");
    }
} while (opcao != "9");
console.log("Aplicação encerrada");
```

ely.miranda@ifpi.edu.br

19

19



Programação Orientada a Objetos

Exceções – Parte 2

Ely – ely.miranda@ifpi.edu.br

20