Residência em Software

Residência em Tecnologia da Informação e Comunicação

Tratamento de Ecxeções em Java

Professor:

Alvaro Degas Coelho

Java

INSTITUIÇÃO EXECUTORA

cepedi UESC

COORDENADORA

MCTI FUTURO Softex

APOIO

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

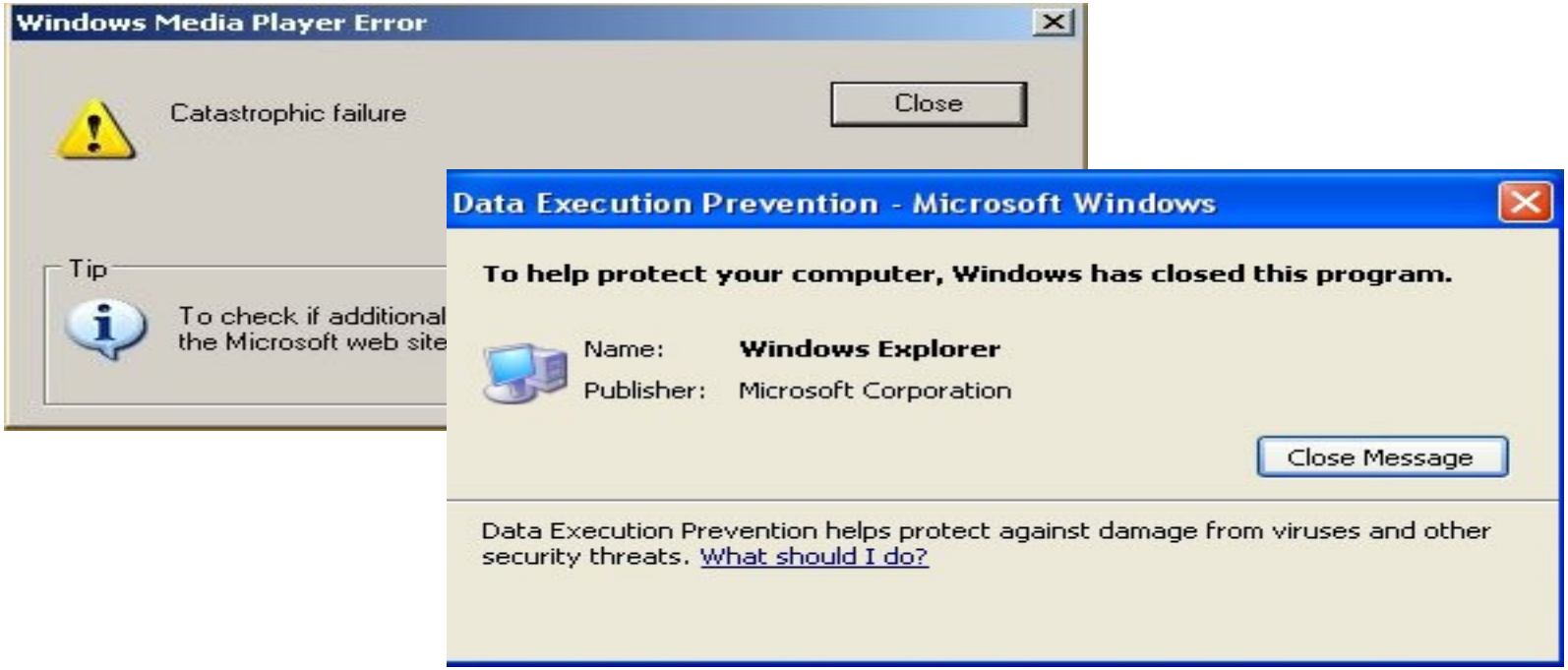
Residência em Software

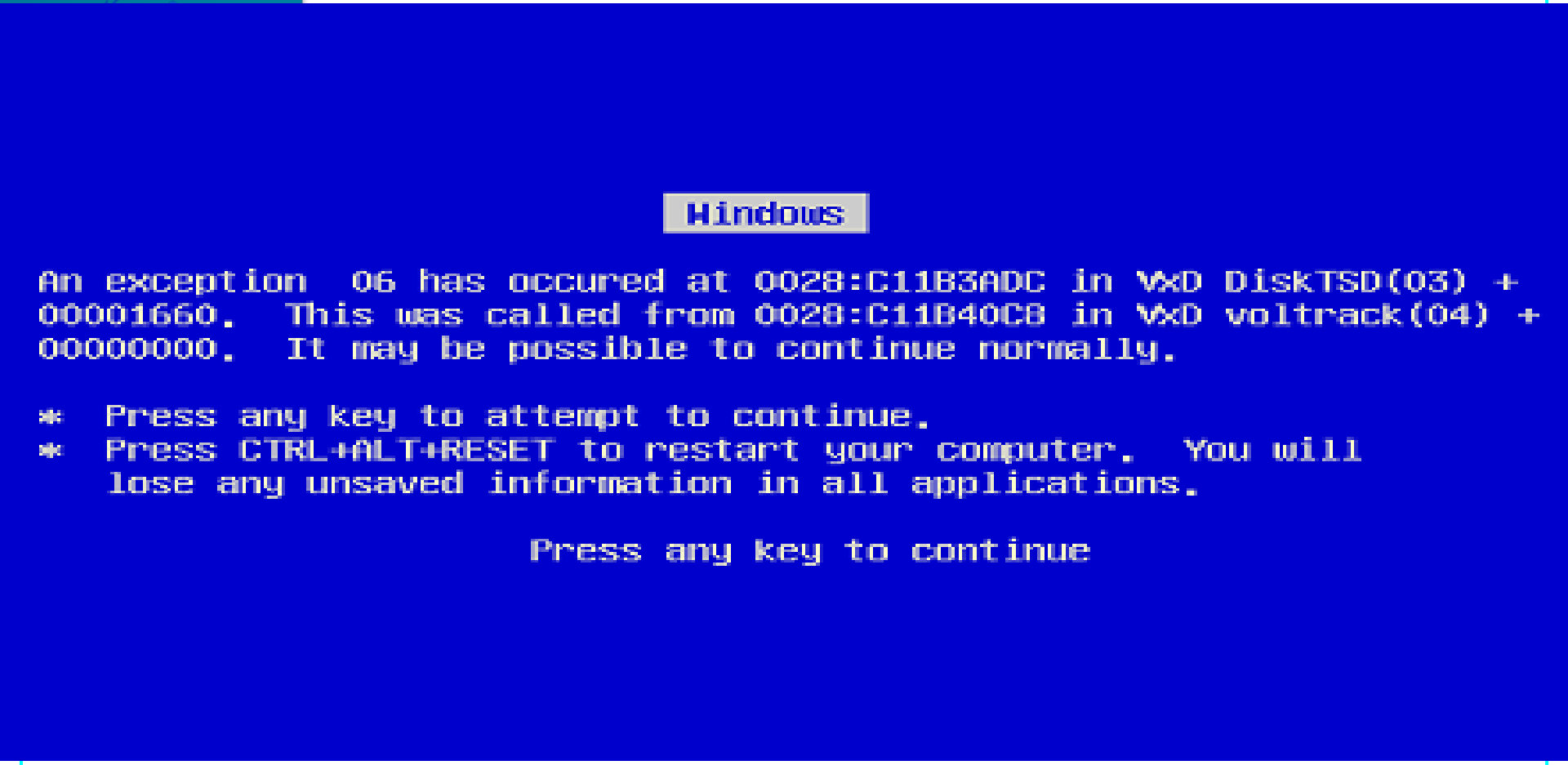
Definição

- Evento que ocorre durante a execução de um programa, que interrompe o fluxo normal de instruções

Situações excepcionais

- Como a Microsoft trata?




Windows
An exception 06 has occured at 0028:C11B3ADC in VxD DiskTSD(03) +
00001660. This was called from 0028:C11B40C8 in VxD voltrack(04) +
00000000. It may be possible to continue normally.
* Press any key to attempt to continue.
* Press CTRL+ALT+RESET to restart your computer. You will
lose any unsaved information in all applications.
Press any key to continue

Residência em Software

Erros em Java

- Quando acontece uma situação excepcional na execução de um programa Java, como é feita a notificação?

Residência em Software

Erros em Java

- A Api de Java define uma classe Throwable, que define todos os erros e exceções.
- Da classe Throwable, foram criadas as classes Exception e Error.

Residência em Software

A família Error

- A classe Error serve para representar condições anormais, que não deveriam ocorrer nunca.
- Programas não devem tratar Errors.
- Exemplo:

  OutOfMemoryError

  Se você tentar alocar 2147483648 inteiros.

Residência em Software

A família Exception

- A classe Exception representa os erros que um programa deve (ou pode) tratar.
- A partir da classe Exception foi criado mais um subtipo: RuntimeException.

Residência em Software

A família RuntimeException

- São exceções não checadas.
  Mais disso daqui a alguns slides.
- Vocês provavelmente já encontraram sua filha mais famosa!
  NullPointerException
- E a filha menos famosa:
  ArithmeticException

Residência em Software

Exceções

- As exceções predefinidas de Java não servem para todas as situações.
- Nós, programadores, queremos criar nossas próprias exceções.
  - Segurança: esconder aspectos da arquitetura de hardware e, principalmente, software
  - Comunicação mais eficiente com o usuário
    - Melhor receber uma mensagem: “O dado digitado impede que o cálculo seja realizado” do que a mensagem “Arithmetic Exception”
- Precisamos crias nossas próprias Exceptions
  - A que família nossas exceções pertencem?

Residência em Software

Criando exceções

- Uma exceção é uma classe como outra qualquer, vocês já sabem criar uma!
- Devem *estender* Exception.
  - Herança
- Para facilitar a legibilidade, e seguir os padrões, sugere-se que os nomes das Exceptions devem acabar com Exception.

- Dois exemplos:

Residência em Software

Criando exceções

```
public class PepinoException extends Exception {
   /*
          Corpo da exceção.
   */
}
```

- Agora, uma exceção séria.

Criando exceções

```
public class ImpossivelDividirPorZeroException {

   int dividendo;

   public ImpossivelDividirPorZeroException (int dividendo){
   this.dividendo = dividendo;
   }

   public String getMessage() {
   return "Impossível dividir " + dividendo + " por zero!";
   }
}
```

- Isso é uma exceção?

Criando exceções

```
public class ImpossivelDividirPorZeroException extends   Exception {

   int dividendo;

   public ImpossivelDividirPorZeroException (int dividendo){
   this.dividendo = dividendo;
   }

   public String getMessage() {
   return "Impossível dividir " + dividendo + " por zero!";
   }
}
```

- Isso é uma exceção?

Lançando exceções

- Um método que queira lançar uma exceção deverá ter duas coisas a mais.
  1. Declarar no seu cabeçalho que pode lançar uma exceção.
  2. Ao detectar um erro, lançar a exceção.

- Vejamos:

Lançando exceções

```
//...
//...

public int dividir(int dividendo, int divisor)
                    throws ImpossivelDividirPorZeroException {
   if(divisor == 0) {
       throw new ImpossivelDividirPorZeroException(dividendo);
   } else {
       return dividendo / divisor;
   }
```

Lançando exceções

- Observação:

  Um método que chama um outro método que pode lançar uma exceção PRECISA declarar no cabeçalho a possibilidade do lançamento, apesar de não ter o `throw` no seu corpo.

Residência em Software

Exceções checadas e não checadas

- Exceção não checada

  É uma exceção que não precisa ser declarada no cabeçalho do método que a lança.

  Não é “lançada”

Residência em Software

Tratando exceções

- Java nos coloca a disposição três blocos especiais para o tratamento de exceções: `try` `catch` e `finally`.
- Como usamos? No próximo slide!

Residência em Software

Tratando exceções

```
public static void main (String[] args) {
   int a = 10;
   int b = 0;

   try {
   int resultado = dividir(a, b);
   } catch (ImpossivelDividirPorZeroException e) {
   System.out.println(e.getMessage());
   } finally {
   System.out.println("Passei pelo try ou pelo catch");
   }
   System.out.println("O resultado foi: " + resultado);
}
```

Residência em Software

Por que usar exceções?

- Separa o código de tratamento do código “normal”.
- Propagação de erros mais efetiva.
- Possibilitar a criação de diferentes tipos de erros para diferentes situações.