# Paradigmas de Linguagens de Programação

## Manipulação de Exceções

Cristiano Lehrer, M.Sc.





ybadoo

# Introdução à Manipulação de Exceções (1/2)

- Em uma linguagem sem manipulação de exceção:
  - Quando ocorre uma exceção, o controle passa para o sistema operacional, onde uma mensagem será exibida e o programa é encerado.


- Em uma linguagem com manipulação de exceção:
  - Os programas são autorizados a capturar algumas exceções, proporcionando assim a possibilidade de corrigir o problema e continuar a sua execução.





ybadoo

# Introdução à Manipulação de Exceções (2/2)

- Muitas linguagens de programação permitem aos programas capturarem as exceções de entrada/saída, como por exemplo, fim de arquivo (*end-of-file* – `EOF`):
  - FORTRAN:
    - `READ(UNIT=5, FMT=1000, ERR=100, END=999) WEIGHT`
      - `ERR` – linha de desvio caso ocorra uma exceção
      - `END` – linha de desvio caso ocorra o fim do arquivo
- Algumas linguagens de programação permitem aos programas capturarem exceções lógicas, como por exemplo, validação do subscrito de *arrays*:
  - Java:

```
int[] array = new int[5];
array[7] = 3; // ArrayIndexOutOfBoundsException
```

# Conceitos Básicos (1/3)

- Uma exceção é qualquer evento, errôneo ou não, que seja detectável por hardware ou software e que possa exigir processamento especial.
- O processamento especial que pode ser exigido pela detecção de uma exceção é chamada manipulação de exceção.
- Esse processamento é feito por uma unidade de código chamada manipulador de exceção.
- Uma exceção é gerada quando ocorre seu evento associado.

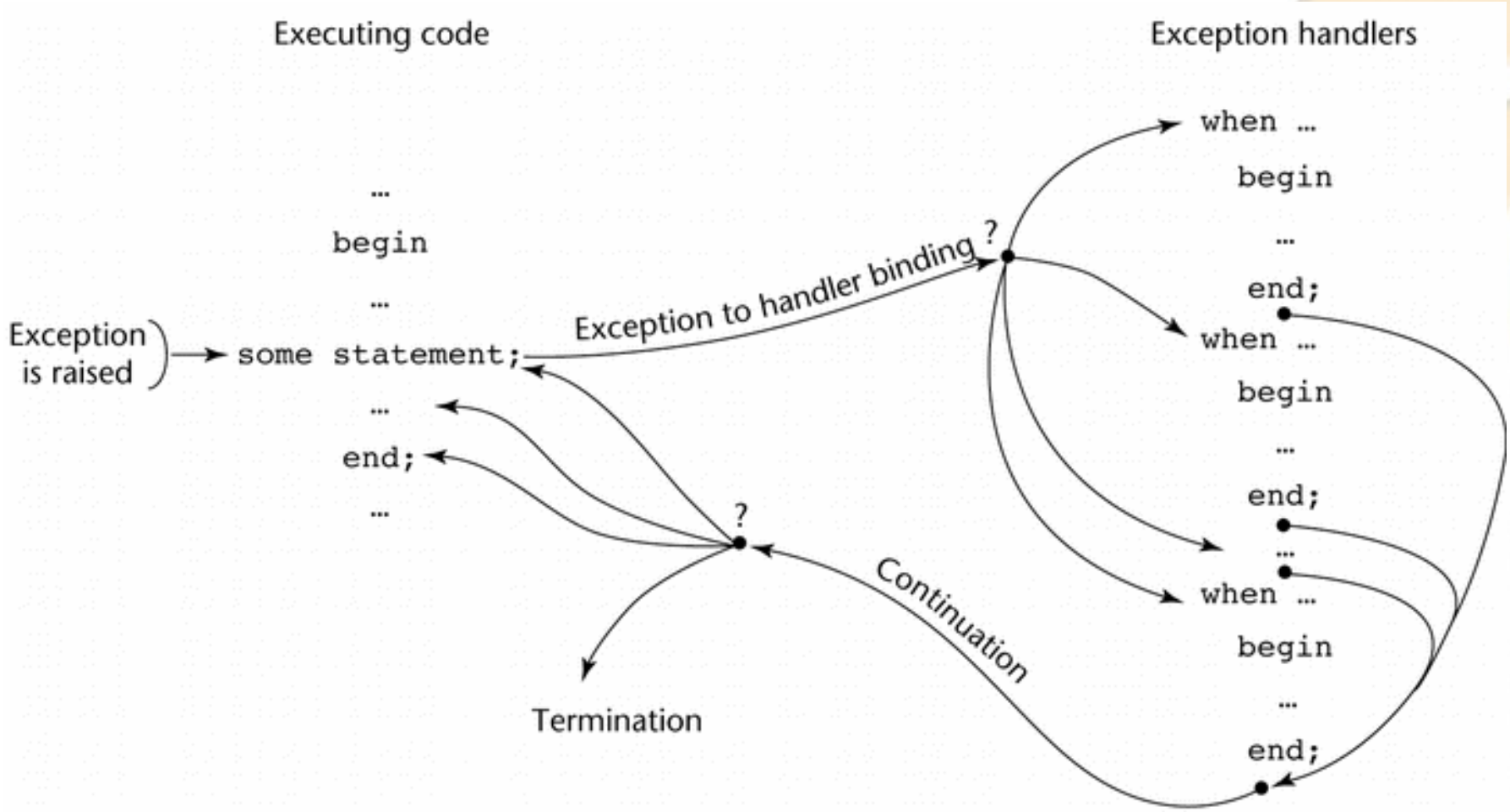
# Conceitos Básicos (2/3)

- Uma linguagem que não possua as capacidades de manipulação de exceções pode ainda definir, identificar, gerar e lidar com exceções definidas pelo usuário ou produzidas pelo compilador.
- Alternativas:
  - Enviar um parâmetro auxiliar ou usar o valor de retorno para indicar o status de retorno de um subprograma.
    - C:
      - `int printf(const char *format, ...);`
        - Um valor de retorno positivo significa o número de caracteres realmente escritos.
        - Um valor de retorno negativo indica um erro.
  - Passar um rótulo (*label*) como parâmetro ao subprograma.
  - Ter o manipulador como um subprograma separado passado como um parâmetro à unidade chamada.

# Conceitos Básicos (3/3)

- Vantagens da manipulação de exceções incorporada a uma linguagem:
  - O código de erro de detecção é entediante de se escrever, além de complicar o código-fonte do programa.
    - C++:
      - `double media = (quantidade != 0) : soma / quantidade ? 0;`
  - Propagação de exceções permitem um elevado nível de reutilização de manipuladores de exceção.

# Fluxo de Controle da Manipulação de Exceções



# Manipulação de Exceções em C++ (1/6)

- Adicionado ao C++ em 1990.
- O projeto de manipulação de exceções é baseado, parcialmente, da CLU, da Ada e da ML.
- Forma geral da construção de manipuladores de exceção:

```
try {
  //** código que é esperado para gerar uma exceção
}
catch(parâmetro formal) {
  //** corpo do manipulador de exceções
}
catch(parâmetro formal) {
  //** corpo do manipulador de exceções
}
```





ybadoo

# Manipulação de Exceções em C++ (2/6)

- `catch` são manipuladores de exceção:
  - É um nome sobrecarregado, portanto, o parâmetro formal de cada `catch` deve ser único.
- O parâmetro formal não precisa ter uma variável.
- Pode ser simplesmente um tipo simples, para distinguir o manipulador dos outros.
- O parâmetro formal pode ser usado para transferir informações para o manipulador.
- O parâmetro formal pode ser uma reticências (`...`), no qual se torna um manipulador pega-tudo, ou seja, lida com todas as exceções ainda não tratadas.

```
try
{ }
catch(int i)
{ }
catch(float f)
{ }
catch(...)
{ }
```





ybadoo

# Manipulação de Exceções em C++ (3/6)

- Vinculando exceções a manipuladores:
  - As exceções são geradas explicitamente pela instrução:
    - **`throw`** `[expressão]`
  - Os colchetes são meta-símbolos usados para especificar que a expressão é opcional.
  - Uma `throw` sem um operando só pode aparecer em um manipulador:
    - Quando aparece nele, gera a exceção, a qual é manipulada em outro lugar.
  - A palavra `throw` foi escolhida porque tanto `signal` como `raise` são funções na biblioteca de padrões ANSI C.
  - Uma exceção sem tratamento é propagada ao chamador da função em que a mesma foi originada:
    - Essa propagação continua até a função principal (`main`).
    - Se nenhum manipulador for encontrado, o programa é encerrado.

# Manipulação de Exceções em C++ (4/6)

- Continuação:
  - Depois que um manipulador concluiu sua execução, o controle flui para a primeira instrução depois da construção `try`.
  - Um manipulador pode gerar novamente uma exceção, usando uma `throw` sem uma expressão, em cujo caso é propagada para o chamador.

```
int main () {
  char myarray[10];
  try {
    for (int n=0; n<=10; n++) {
      if (n>9) throw "Fora do limite";
      myarray[n]='z';
    }
  }
  catch (char * str) {
    cout << "Exceção: " << str << endl;
  }
  return 0;
}
```

# Manipulação de Exceções em C++ (5/6)

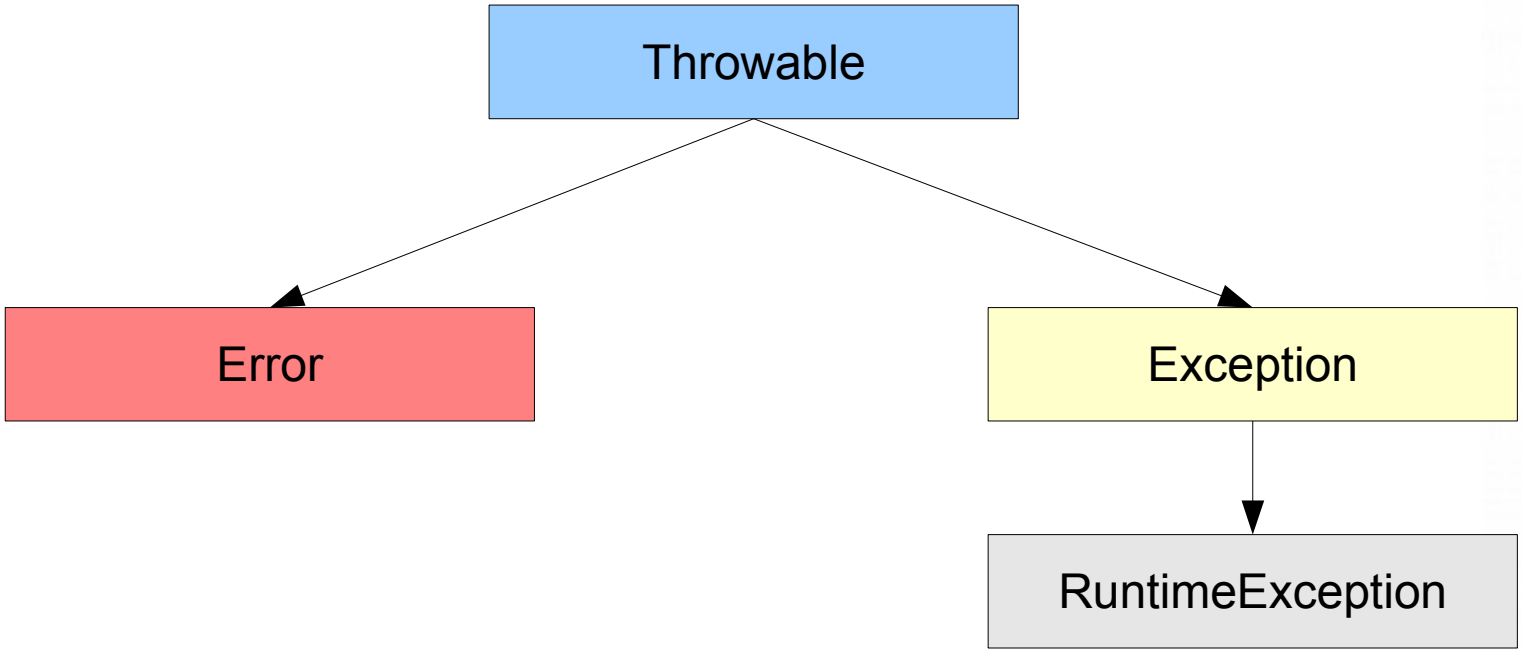
- Outras opções de projeto:
  - Todas as exceções são definidas pelo usuário.
  - Exceções não são especificadas:
    - Ainda que possam ser declaradas como novas classe.
  - Não há manipuladores padrão:
    - Porque as exceções detectadas pelo sistema não podem ser manipuladas.
  - Exceções não podem ser desativadas.
  - As funções podem listar os tipos de exceções que podem gerar
    - `int fun() throw (int, char *( {…}`
  - Se não houver nenhuma especificação `throw` no cabeçalho, a função poderá gerar qualquer exceção.

# Manipulação de Exceções em C++ (6/6)

- Avaliação:
  - É estranho que as exceções não sejam nomeadas, e que as exceções de hardware e software do sistema não sejam detectáveis, impossibilitando o tratamento.
  - A vinculação da exceção com o tipo do parâmetro formal certamente não promove a legibilidade:
    - É muito melhor definir classes com nomes significativos em uma hierarquia significativa que possam ser usadas para definir exceções.

# Manipulação de Exceções em Java (1/10)

- A manipulação de exceções do Java baseia-se na do C++, mas foi projetada para estar mais próxima do paradigma das linguagens orientadas a objeto.
- Todas as exceções são objetos de classes que são descendentes da classe `Throwable`.








ybadoo

# Manipulação de Exceções em Java (2/10)

- A biblioteca Java inclui duas subclasses de `Throwable`:
  - `Error`:
    - Erros gerados pelo interpretador Java, como por exemplo, o esgotamento de memória do *heap*.
    - Jamais serão geradas por programas usuários e nunca devem ser manipuladas.
  - `Exception`:
    - Exceções definidas pelo usuário são subclasses dessa.
    - `RuntimeException` – geradas quando um programa usuário causa um erro, mas com o tratamento opcional.
    - `IOException` – geradas quando ocorre um erro em uma operação de entrada e saída.

# Manipulação de Exceções em Java (3/10)

- Manipuladores de exceção:
  - Têm a mesma forma que os do C++, exceto:
    - O parâmetro de cada `catch` deve estar presente.
    - Sua classe deve ser uma descendente da classe `Throwable`.
  - A sintaxe da cláusula `try` é exatamente a do C++.
  - Exceções são lançadas com `throw`, como em C++, mas frequentemente com o operador `new`, para criar um objeto:
    - `throw new MyException();`

# Manipulação de Exceções em Java (4/10)

- A vinculação de exceções a manipuladores em Java é menos complexa do que no C++.
- Uma exceção é vinculada ao primeiro manipulador cujo parâmetro é da mesma classe que o objeto gerado ou um ancestral dele.
- Uma exceção podem ser manipuladas e depois regeradas, incluindo uma instrução `throw` sem um operando no final do manipulador:
  - O manipulador também pode gerar uma exceção diferente.

# Manipulação de Exceções em Java (5/10)

- Continuação:
  - Se nenhum manipulador for encontrado num `try`, a busca continua nos manipuladores do próximo `try`, e assim por diante.
  - Se nenhum manipulador puder ser encontrado no método, a exceção é propagada para o método chamador, até chegar ao método `main`.
  - Se nenhum manipulador for encontrado, o programa é encerrado.
  - Para garantir que todas as exceções sejam capturadas, pode ser incluído no `try` um manipulador para capturar todas as exceções:
    - Simplesmente use uma classe `Exception` como parâmetro.
    - Esse manipulador deverá ser o último num `try`.

# Manipulação de Exceções em Java (6/10)

- Outras opções de projeto:
  - A cláusula `throws` em Java é muito diferente da cláusula `throw` em C++.
  - Exceções das classes `Error` e `RuntimeException` e todas as suas descendentes são chamadas de exceções não-verificadas.
  - Todas as outras exceções são chamadas de exceções verificadas.
  - Um método com exceções verificadas deve assegurar:
    - Que elas estão listadas em sua cláusula `throws`, ou
    - Manipuladas no método.
  - Um método herdado não pode declarar mais exceções na sua cláusula `throws` do que o método original.





ybadoo

# Manipulação de Exceções em Java (7/10)

- Um método que chama um método que lista uma exceção, marcada na sua cláusula `throws` tem três alternativas com essa exceção:
  - Capturar e tratar a exceção.
  - Capturar a exceção e lançar uma exceção que esteja listada na sua própria cláusula `throws`.
  - Declará-la na sua cláusula `throws` e não manipulá-lo.

# Manipulação de Exceções em Java (8/10)

- A cláusula `finally`:
  - Pode aparecer no fim de uma estrutura `try`.
  - Propósito:
    - Especificar o código que será executado, independentemente da geração de uma exceção ou não pelo bloco protegido.

```
try {

}
catch(...) {

}
catch(...) {

}
finally {

}
```





ybadoo

# Manipulação de Exceções em Java (9/10)

- Uma construção `try` com `finally` pode ser utilizada sem a especificação de nenhum manipulador de exceções (`catch`).

```
try
{
   for (index = 0; index < 100; index++)
   {
      ...
      if (...)
      {
         return;
      }
   }
}
finally
{
   ...
}
```





ybadoo

# Manipulação de Exceções em Java (10/10)

- Avaliação:
  - Os tipos de exceções fazem mais sentido do que no caso do C++.
  - A cláusula `throws` é melhor do que a utilizada pelo C++:
    - A cláusula `throw` em C++ diz pouco ao programador.
  - A cláusula `finally` é frequentemente útil.
  - O interpretador Java gera uma série de exceções que podem ser manipulados pelos programas do usuário.