

Programação Orientada a Objetos

<http://java.icmc.usp.br/moodle/>

POO 15: Tratamento de Exceções

PAE: Pedro Shiguihara-Juárez
Professor: Dilvan de Abreu Moreira

Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo

2012-06-12

Leitura

- Tratamento de Exceções: Responder a situações de erros que surgem na execução de um programa.

Leitura

- Tratamento de Exceções: Responder a situações de erros que surgem na execução de um programa.
- Quais tópicos serão cobertos nesta aula?

Outline

- 1 Tipos de erros
- 2 Erro lógico
- 3 Caso: Calculadora
- 4 Exceções
- 5 Assertivas

Tipos de erros

```
1 //Soma
2 double temp = 15;
3 temp = temp + 4.0;
4 System.out.println(temp);
```

```
1 //Divisão
2 double res;
3 Scanner scan;
4 scan = new Scanner(System.in);
5 double num1 = scan.nextDouble();
6 double num2 = scan.nextDouble();
7
8 res = num1 / num2;
9 System.out.println(res);
```

Tipos de erros

```
1 //Soma
2 double temp = 15;
3 temp = temp + 4.0
4 System.out(temp);
```

Erro sintático.

```
1 //Divisão
2 double res;
3 Scanner scan;
4 scan = new Scanner(System.in);
5 double num1 = scan.nextDouble();
6 double num2 = scan.nextDouble();
7
8 res = num1 / num2;
9 System.out.println(res);
```

Erro lógico.

Tipos de erros

```
1 //Soma
2 double temp = 15;
3 temp = temp + 4.0
4 System.out(temp);
```

Erro sintático.

```
1 //Divisão
2 double res;
3 Scanner scan;
4 scan = new Scanner(System.in);
5 double num1 = scan.nextDouble();
6 double num2 = scan.nextDouble();
7
8 res = num1 / num2;
9 System.out.println(res);
```

Erro lógico.

- Erros lógicos são mais difíceis de identificar.

Tipos de erros

```
1 //Soma
2 double temp = 15;
3 temp = temp + 4.0
4 System.out(temp);
```

Erro sintático.

```
1 //Divisão
2 double res;
3 Scanner scan;
4 scan = new Scanner(System.in);
5 double num1 = scan.nextDouble();
6 double num2 = scan.nextDouble();
7
8 res = num1 / num2;
9 System.out.println(res);
```

Erro lógico.

- Erros lógicos são mais difíceis de identificar.
- Compilador não pode fornecer nenhuma ajuda quanto a erros lógicos.

Outline

- 1 Tipos de erros
- 2 Erro lógico
- 3 Caso: Calculadora
- 4 Exceções
- 5 Assertivas

Algumas causas de erro

- Implementação incorreta.

Algumas causas de erro

- Implementação incorreta.

```
1 //Array ordenado
2 int [] array = { 3,4,6,7,9 };
3
4 //Calcular valor mediano
5 double res = 0;
6 for (int num:array) {
7     res = res+num;
8 }
9 res = res/array.length;
10 System.out.println("Valor mediano: "+res);
```

Algumas causas de erro

- Implementação incorreta.

```
1 //Array ordenado
2 int [] array = { 3,4,6,7,9 };
3
4 //Calcular valor mediano
5 double res = 0;
6 for (int num:array) {
7     res = res+num;
8 }
9 res = res/array.length;
10 System.out.println("Valor mediano: "+res);
```

- Não cumpre a especificação.

Algumas causas de erro

- Implementação incorreta.

```
1 //Array ordenado
2 int [] array = { 3,4,6,7,9 };
3
4 //Calcular valor mediano
5 double res = 0;
6 for (int num:array) {
7     res = res+num;
8 }
9 res = res/array.length;
10 System.out.println("Valor mediano: "+res);
```

- Não cumpre a especificação.
- Calcula valor medio em vez de valor mediano

Algumas causas de erro

- Implementação incorreta.

```
1 //Array ordenado
2 int [] array = { 3,4,6,7,9 };
3
4 //Calcular valor mediano
5 double res = 0;
6 for (int num:array) {
7     res = res+num;
8 }
9 res = res/array.length;
10 System.out.println("Valor mediano: "+res);
```

- Não cumpre a especificação.
- Calcula valor medio em vez de valor mediano
- Estado de objeto inconsistente ou impróprio.

Algumas causas de erro

- Solicitação de objeto inapropriada.

```
1 //Variável idades
2 int[] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```

IDADES

→

1

2

4

11

31

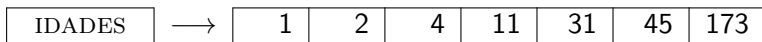
45

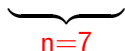
173

Algumas causas de erro

- Solicitação de objeto inapropriada.

```
1 //Variável idades
2 int [] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```

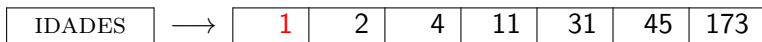



n=7

Algumas causas de erro

- Solicitação de objeto inapropriada.

```
1 //Variável idades
2 int [] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```



$n=7$

$i=0$

Algumas causas de erro

- Solicitação de objeto inapropriada.

```
1 //Variável idades
2 int [] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```



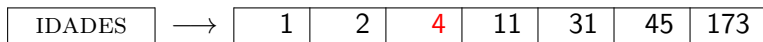
$\underbrace{\hspace{10em}}$
n=7

i=0 i=1

Algumas causas de erro

- Solicitação de objeto inapropriada.

```
1 //Variável idades
2 int [] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```



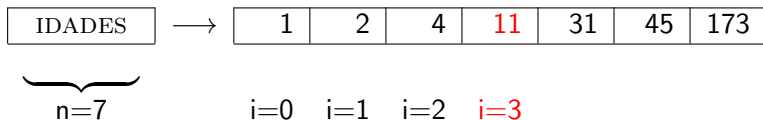
$\underbrace{\hspace{10em}}$
n=7

i=0 i=1 i=2

Algumas causas de erro

- Solicitação de objeto inapropriada.

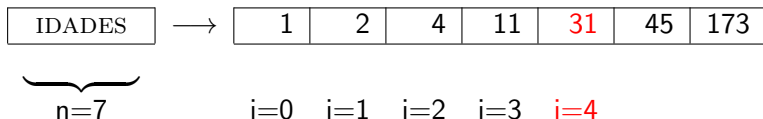
```
1 //Variável idades
2 int [] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```



Algumas causas de erro

- Solicitação de objeto inapropriada.

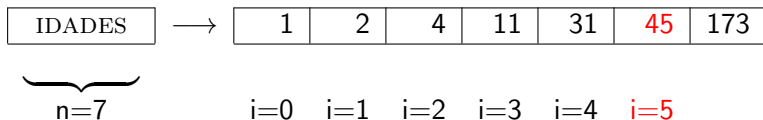
```
1 //Variável idades
2 int [] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```



Algumas causas de erro

- Solicitação de objeto inapropriada.

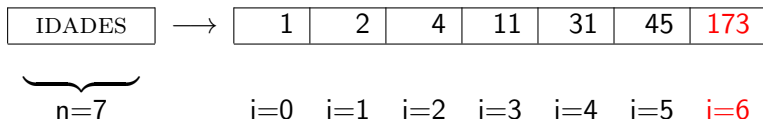
```
1 //Variável idades
2 int [] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```



Algumas causas de erro

- Solicitação de objeto inapropriada.

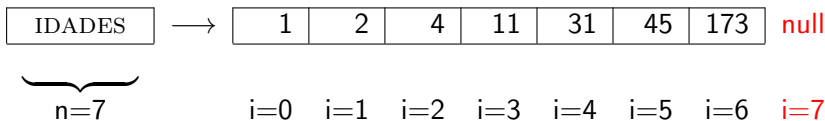
```
1 //Variável idades
2 int [] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```



Algumas causas de erro

- Solicitação de objeto inapropriada.

```
1 //Variável idades
2 int [] idades = {1, 2, 4, 11, 31, 45, 173};
3
4 int n = idades.length;
5 for (int i=0; i<=n; i++)
6 {
7     System.out.println( idades[i] );
8 }
```



Algumas causas de erro

- Em geral, os erros surgem do ambiente:
 - URL incorreto.
 - Interrupção de rede.
- Processamento de arquivo é propenso a erro particular
 - Falta de arquivos.
 - Falta de permissões apropriadas.

Outline

- 1 Tipos de erros
- 2 Erro lógico
- 3 Caso: Calculadora**
- 4 Exceções
- 5 Assertivas

Caso: Calculadora

- Exploramos situações de erro por meio do programa “Calculadora”.

Caso: Calculadora

- Exploramos situações de erro por meio do programa “Calculadora”.
- Uma calculadora é um objeto típico de serviço (programa servidor).

Caso: Calculadora

- Exploramos situações de erro por meio do programa “Calculadora”.
- Uma calculadora é um objeto típico de serviço (programa servidor).
- Todas as suas atividades estão baseadas nas solicitações do cliente.

Dois aspectos:

- Informe de erro.
- Tratamento de erros.

Informe de erro

Algumas questões:

- Um programa deve assumir que os clientes são bem-comportados?
- Ou deve assumir que são potencialmente hostis?

Informe de erro

Algumas questões:

- Quantas verificações um programa faz a cada chamada de método?
- Como relatar erros?
- Como um cliente pode antecipar a falha?
- Como um cliente deve lidar com a falha?

Informe de erro

- Divisão por zero.

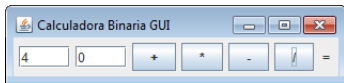
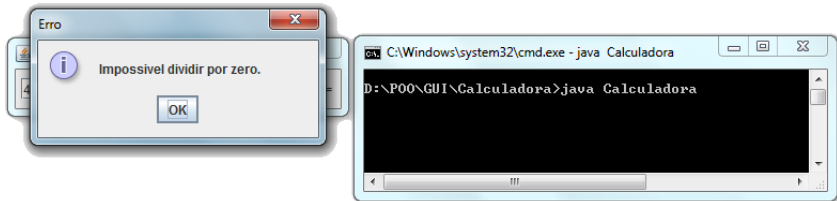


Figura : Programa *Calculadora*



Informe de erro

- Argumentos representam uma 'vulnerabilidade'.

Informe de erro

- Argumentos representam uma ‘vulnerabilidade’.
- Os argumentos de método frequentemente contribuem para o comportamento.

Informe de erro

- Argumentos representam uma ‘vulnerabilidade’.
- Os argumentos de método frequentemente contribuem para o comportamento.
- Argumentos de construtor inicializam o estado.

Informe de erro

- Argumentos representam uma ‘vulnerabilidade’.
- Os argumentos de método frequentemente contribuem para o comportamento.
- Argumentos de construtor inicializam o estado.
- Verificação de argumento é uma medida defensiva.

Informe de erro

- O que acontece ao usar “4,5” em vez de “4.5”?



```
C:\Windows\system32\cmd.exe - java Calculadora
D:\POO\GUI\Calculadora>java Calculadora
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "4,5"
    at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1241)
    at java.lang.Double.parseDouble(Double.java:540)
    at Calculadora$4.actionPerformed(Calculadora.java:113)
    at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:2018)
    at javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2341)
    at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:402)
    at javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:259)
    at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:252)
    at java.awt.Component.processMouseEvent(Component.java:6505)
    at javax.swing.JComponent.processMouseEvent(JComponent.java:3321)
    at java.awt.Component.processEvent(Component.java:6270)
    at java.awt.Container.processEvent(Container.java:2229)
    at java.awt.Component.dispatchEventImpl(Component.java:4861)
    at java.awt.Container.dispatchEventImpl(Container.java:2287)
    at java.awt.Component.dispatchEvent(Component.java:4687)
    at java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4032)
```

Informe de erro

- O que acontece ao usar “4,5” em vez de “4.5”?



```
C:\Windows\system32\cmd.exe - java Calculadora
D:\P00\GUI\Calculadora>java Calculadora
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "4,5"
    at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1241)
    at java.lang.Double.parseDouble(Double.java:540)
    at Calculadora$4.actionPerformed(Calculadora.java:113)
    at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:2018)
    at javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2341)
    at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:402)
    at javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:259)
    at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:252)
    at java.awt.Component.processMouseEvent(Component.java:6505)
    at javax.swing.JComponent.processMouseEvent(JComponent.java:3321)
    at java.awt.Component.processEvent(Component.java:6270)
    at java.awt.Container.processEvent(Container.java:2229)
    at java.awt.Component.dispatchEventImpl(Component.java:4861)
    at java.awt.Container.dispatchEventImpl(Container.java:2287)
    at java.awt.Component.dispatchEvent(Component.java:4687)
    at java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4032)
```

- O resultado é um erro de runtime.

Informe de erro

- O que acontece ao usar "4,5" em vez de "4.5"?



```
C:\Windows\system32\cmd.exe - java Calculadora
D:\P00\GUI\Calculadora>java Calculadora
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "4,5"
    at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1241)
    at java.lang.Double.parseDouble(Double.java:540)
    at Calculadora$4.actionPerformed(Calculadora.java:113)
    at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:2018)
    at javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2341)
    at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:402)
    at javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:259)
    at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:252)
    at java.awt.Component.processMouseEvent(Component.java:6505)
    at javax.swing.JComponent.processMouseEvent(JComponent.java:3321)
    at java.awt.Component.processEvent(Component.java:6270)
    at java.awt.Container.processEvent(Container.java:2229)
    at java.awt.Component.dispatchEventImpl(Component.java:4861)
    at java.awt.Container.dispatchEventImpl(Container.java:2287)
    at java.awt.Component.dispatchEvent(Component.java:4687)
    at java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4032)
```

- O resultado é um erro de runtime.
- De quem é essa 'falha'?

Informe de erro

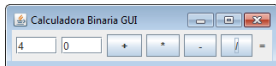
- O que acontece ao usar “4,5” em vez de “4.5”?



```
C:\Windows\system32\cmd.exe - java Calculadora
D:\POO\GUI\Calculadora>java Calculadora
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "4,5"
    at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1241)
    at java.lang.Double.parseDouble(Double.java:540)
    at Calculadora$4.actionPerformed(Calculadora.java:113)
    at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:2018)
    at javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2341)
    at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:402)
    at javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:259)
    at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:252)
    at java.awt.Component.processMouseEvent(Component.java:6505)
    at javax.swing.JComponent.processMouseEvent(JComponent.java:3321)
    at java.awt.Component.processEvent(Component.java:6270)
    at java.awt.Container.processEvent(Container.java:2229)
    at java.awt.Component.dispatchEventImpl(Component.java:4861)
    at java.awt.Container.dispatchEventImpl(Container.java:2287)
    at java.awt.Component.dispatchEvent(Component.java:4687)
    at java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4032)
```

- O resultado é um erro de runtime.
- De quem é essa 'falha'?
- Antecipação e prevenção são preferíveis a dividir a culpa.

Valores de argumento



```
1 double valor1 = Double.parseDouble(textValor1);
2 double valor2 = Double.parseDouble(textValor2);
3
4 if (valor2 == 0){
5     showErroDivisao();
6 } else {
7     double result = valor1 / valor2;
8     total.setText(" = "+result);
9 }
```

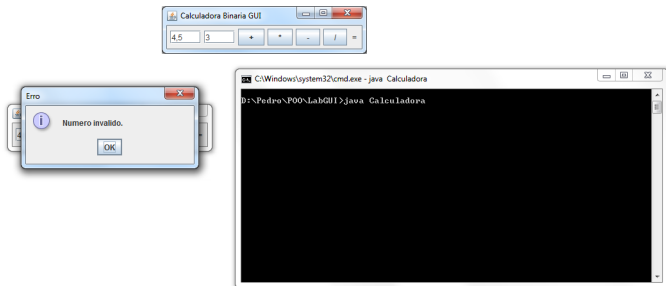
Falta validar se as entradas são números.

```
1 double valor1;
2 double valor2;
3 double result;
4
5 try {
6     valor1 = Double.parseDouble(textValor1);
7     valor2 = Double.parseDouble(textValor2);
8     result = valor1 / valor2;
9     total.setText(" = "+result);
10 } catch ( NumberFormatException ex1 ){
11     showErrorNumberFormat ();
12 } catch ( ArithmeticException ex2 ){
13     showErrorDivideByZero ();
14 }
```

Usando exceções fazemos cobertura de diferentes erros.

Valores de argumento

Usando exceções:



Valores de argumento

```
errorCodeType readFile {
    initialize errorCode = 0;
    open the file;
    if (theFileIsOpen) {
        determine the length of the file;
        if (gotTheFileLength) {
            allocate that much memory;
            if (gotEnoughMemory) {
                read the file into memory;
                if (readFailed) {
                    errorCode = -1;
                }
            } else {
                errorCode = -2;
            }
        } else {
            errorCode = -3;
        }
        close the file;
    }
    ...
}
```

Usando if-else

```
readFile {
    try {
        open the file;
        determine its size;
        allocate that much memory;
        read the file into memory;
        close the file;
    } catch (fileOpenFailed) {
        doSomething;
    } catch (sizeDeterminationFailed) {
        doSomething;
    } catch (memoryAllocationFailed) {
        doSomething;
    } catch (readFailed) {
        doSomething;
    } catch (fileCloseFailed) {
        doSomething;
    }
}
```

Usando exceções

Outline

- 1 Tipos de erros
- 2 Erro lógico
- 3 Caso: Calculadora
- 4 Exceções**
- 5 Assertivas

Definição de Exceção

Uma exceção é uma indicação de um problema que ocorre durante a execução de um programa. Com o tratamento de exceções, um programa pode continuar executando, em vez de encerrar, depois de lidar com o problema. Isto permite escrever programas robustos e tolerantes a falhas.

Definição de Exceção

Uma **exceção** é uma indicação de um problema que ocorre durante a execução de um programa. Com o tratamento de exceções, um programa pode continuar executando, em vez de encerrar, depois de lidar com o problema. Isto permite escrever programas robustos e tolerantes a falhas.

Definição de Exceção

Uma exceção é uma indicação de um problema que ocorre **durante a execução** de um programa. Com o tratamento de exceções, um programa pode continuar executando, em vez de encerrar, depois de lidar com o problema. Isto permite escrever programas robustos e tolerantes a falhas.

Definição de Exceção

Uma exceção é uma indicação de um problema que ocorre durante a execução de um programa. Com o tratamento de exceções, **um programa pode continuar executando**, em vez de encerrar, depois de lidar com o problema. Isto permite escrever programas robustos e tolerantes a falhas.

Definição de Exceção

Uma exceção é uma indicação de um problema que ocorre durante a execução de um programa. Com o tratamento de exceções, um programa pode continuar executando, em vez de encerrar, depois de lidar com o problema. Isto permite escrever **programas robustos e tolerantes a falhas**.

Definição de Exceção

Uma **exceção** é uma indicação de um problema que ocorre **durante a execução** de um programa. Com o tratamento de exceções, **um programa pode continuar executando**, em vez de encerrar, depois de lidar com o problema. Isto permite escrever **programas robustos e tolerantes a falhas**.

Princípios de lançamento de exceção

- Nenhum valor de retorno “especial” necessário.

Princípios de lançamento de exceção

- Nenhum valor de retorno “especial” necessário.
- Não é possível ignorar os erros no cliente.

Princípios de lançamento de exceção

- Nenhum valor de retorno “especial” necessário.
- Não é possível ignorar os erros no cliente.
 - O controle de fluxo normal é interrompido.

Princípios de lançamento de exceção

- Nenhum valor de retorno “especial” necessário.
- Não é possível ignorar os erros no cliente.
 - O controle de fluxo normal é interrompido.
- Ações de recuperação específicas são incentivadas. Isto, porque só a decisão de lançar uma exceção evitará ativamente que o cliente ignore as consequências da falha do método.

Princípios de lançamento de exceção

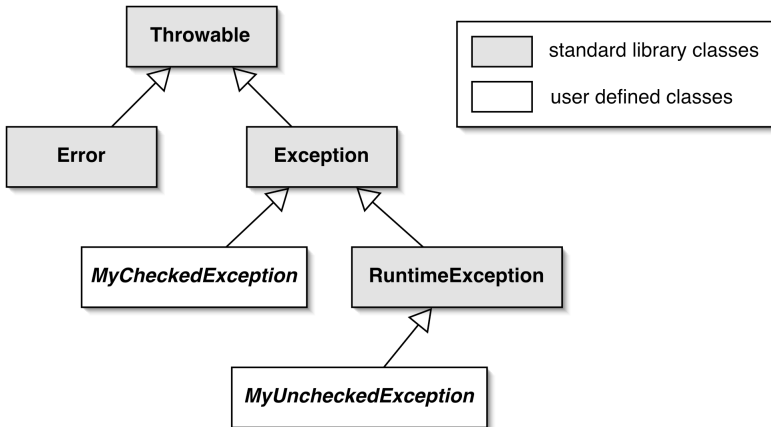
- Nenhum valor de retorno “especial” necessário.
- Não é possível ignorar os erros no cliente.
 - O controle de fluxo normal é interrompido.
- Ações de recuperação específicas são incentivadas. Isto, porque só a decisão de lançar uma exceção evitará ativamente que o cliente ignore as consequências da falha do método.
- O não tratamento de uma exceção pelo cliente fará com que a aplicação termine imediatamente.

Lançando uma exceção

- Um objeto de exceção é criado.
 - `new ExceptionType(``...``);`
- O objeto de exceção é lançado.
 - `throw ...`
- Documentação Javadoc.
 - `@throws ExceptionType reason`

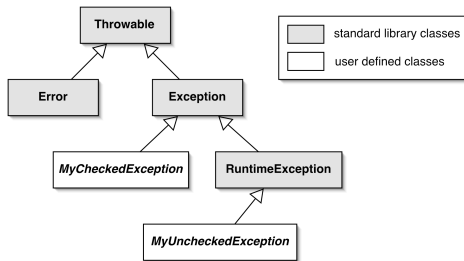
A hierarquia de classes de exceção

Throwable é definida no pacote `java.lang`.



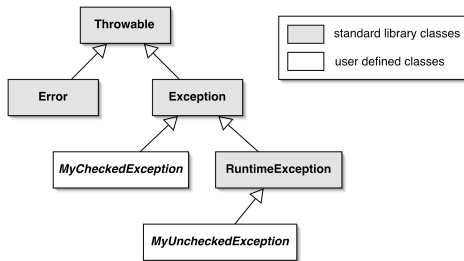
Categoria de exceção

- A classe `Error` e suas subclasses representam situações anormais na JVM.



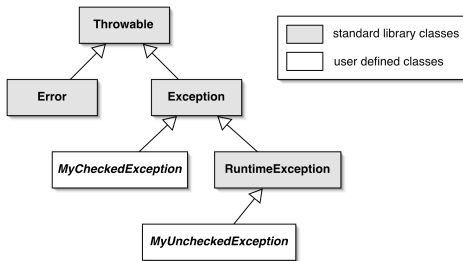
Categoria de exceção

- A classe `Error` e suas subclasses representam situações anormais na JVM.
 - `Errors` não acontecem frequentemente e não devem ser capturados pelos aplicativos.



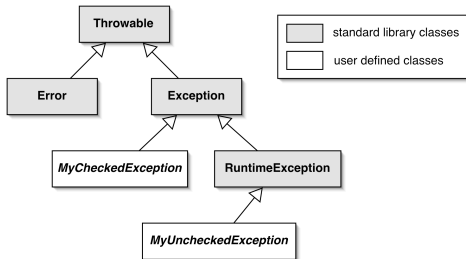
Categoria de exceção

- A classe `Error` e suas subclasses representam situações anormais na JVM.
 - `Errors` não acontecem frequentemente e não devem ser capturados pelos aplicativos.
 - Normalmente não é possível que aplicativos se recuperem de `Errors`.



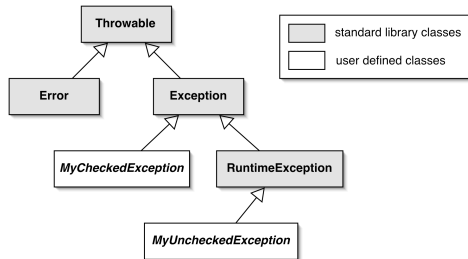
Categoria de exceção

- Exceções não verificadas.



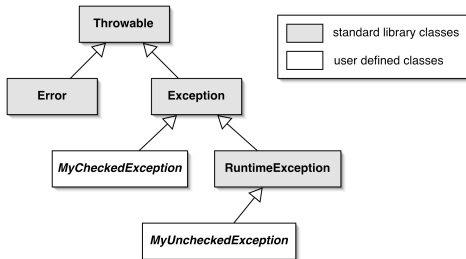
Categoria de exceção

- Exceções não verificadas.
 - Subclasse de `RuntimeException`.



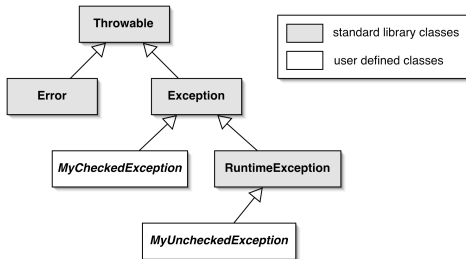
Categoria de exceção

- Exceções não verificadas.
 - Subclasse de `RuntimeException`.
 - Use para falhas antecipadas.



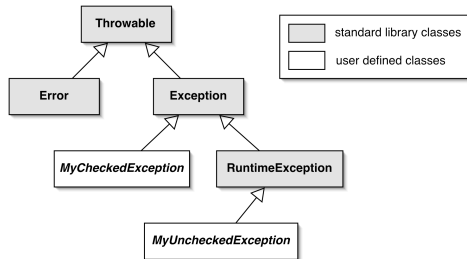
Categoria de exceção

- Exceções não verificadas.
 - Subclasse de `RuntimeException`.
 - Use para falhas antecipadas.
 - Onde a recuperação é improvável.



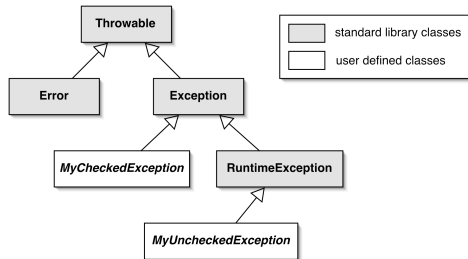
Categoria de exceção

- Exceções não verificadas.
 - Subclasse de `RuntimeException`.
 - Use para falhas antecipadas.
 - Onde a recuperação é improvável.
 - Costumam ser causadas por deficiências no código do seu programa.



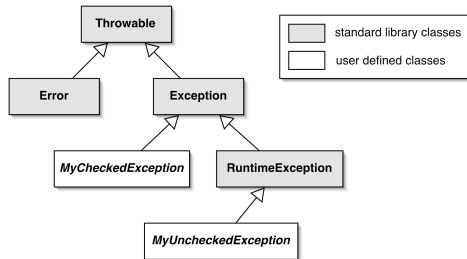
Categoria de exceção

- Exceções não verificadas.
 - Subclasse de `RuntimeException`.
 - Use para falhas antecipadas.
 - Onde a recuperação é improvável.
 - Costumam ser causadas por deficiências no código do seu programa.
- Exceções verificadas.



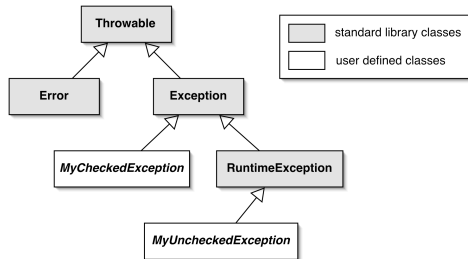
Categoria de exceção

- Exceções não verificadas.
 - Subclasse de `RuntimeException`.
 - Use para falhas antecipadas.
 - Onde a recuperação é improvável.
 - Costumam ser causadas por deficiências no código do seu programa.
- Exceções verificadas.
 - Subclasse de `Exception`.



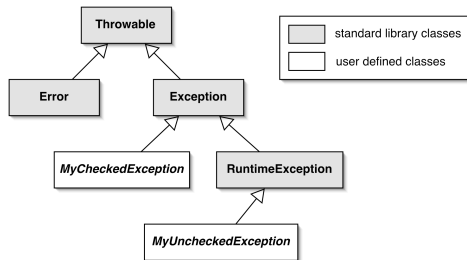
Categoria de exceção

- Exceções não verificadas.
 - Subclasse de `RuntimeException`.
 - Use para falhas antecipadas.
 - Onde a recuperação é improvável.
 - Costumam ser causadas por deficiências no código do seu programa.
- Exceções verificadas.
 - Subclasse de `Exception`.
 - Use para falhas antecipadas.



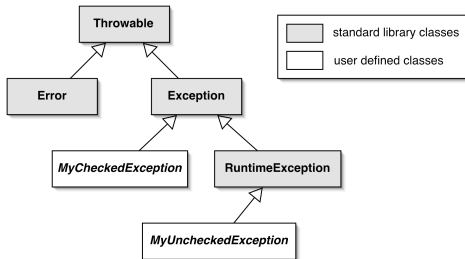
Categoria de exceção

- Exceções não verificadas.
 - Subclasse de `RuntimeException`.
 - Use para falhas antecipadas.
 - Onde a recuperação é improvável.
 - Costumam ser causadas por deficiências no código do seu programa.
- Exceções verificadas.
 - Subclasse de `Exception`.
 - Use para falhas antecipadas.
 - Onde a recuperação é possível.



Categoria de exceção

- Exceções não verificadas.
 - Subclasse de `RuntimeException`.
 - Use para falhas antecipadas.
 - Onde a recuperação é improvável.
 - Costumam ser causadas por deficiências no código do seu programa.
- Exceções verificadas.
 - Subclasse de `Exception`.
 - Use para falhas antecipadas.
 - Onde a recuperação é possível.
 - Tipicamente causadas por condições que não estão no controle do programa.



Exceções não verificadas

- O uso dessa é verificado pelo compilador.
- Faz com que um programa termine se as exceções não são capturadas.
 - Essa é a prática normal.
- `IllegalArgumentException` é um exemplo típico.

Exceções não verificadas

```
1 public String verificarLogin(String usuario){
2     if( usuario == null ){
3         throw new NullPointerException(
4             "null em login");
5     }
6     if( usuario.trim().length() == 0 ){
7         throw new IllegalArgumentException(
8             "login em branco");
9     }
10
11     return usuario;
12 }
```

Exemplo de exceções não verificadas¹.

¹IllegalArgumentException e NullPointerException são parte do pacote `java.lang`.

Exceções verificadas

- Exceções verificadas são projetadas para serem capturadas.

Exceções verificadas

- Exceções verificadas são projetadas para serem capturadas.
- Compilador garante que seu uso é rigorosamente controlado.

Exceções verificadas

- Exceções verificadas são projetadas para serem capturadas.
- Compilador garante que seu uso é rigorosamente controlado.
- Usadas adequadamente, é possível recuperar-se das falhas.

Exceções verificadas

- Os métodos que lançam uma exceção verificada devem incluir uma cláusula throws:
 - `public void saveToFile(String destinationFile) throws IOException`

Exceções verificadas

- Os métodos que capturam uma exceção devem proteger a chamada com uma instrução `try`:

```
try {  
    Proteja uma ou mais instruções aqui.  
}  
catch(Exception e) {  
    Informe e recupere a partir da exceção aqui.  
}
```

Instrução `try`

1. A exceção é lançada daqui

```
try {  
    addressbook.saveToFile(filename);  
    tryAgain = false;  
}  
catch(IOException e) {  
    System.out.println("Unable to save to " + filename);  
    tryAgain = true;  
}
```

2. O controle é transferido para cá

Instrução `try`

Capturando múltiplas exceções:

```
try {
    ...
    ref.process ();
    ...
}
catch (EOFException e) {
    // Entra em ação sobre uma exceção fim de arquivo.
    ...
}
catch (FileNotFoundException e) {
    // Entra em ação sobre uma exceção de arquivo
    // não localizado.
    ...
}
```

Instrução finally

```
try {  
    Proteja uma ou mais instruções aqui.  
}  
catch(Exception e) {  
    Informe e recupere a partir da exceção aqui.  
}  
finally {  
    Realize quaisquer ações comuns aqui se uma  
    exceção for ou não lançada.  
}
```


Instrução `finally`

- Uma cláusula `finally` é executada mesmo que uma instrução de retorno seja executada nas cláusulas `try` ou `catch`.

Instrução `finally`

- Uma cláusula `finally` é executada mesmo que uma instrução de retorno seja executada nas cláusulas `try` ou `catch`.
- Uma exceção não capturada ou propagada ainda existe via a cláusula `finally`.

Tentando recuperação

- Os clientes devem fazer notificações de erro.
- Inclua código para tentar a recuperação.
 - Sempre requerem um loop.

Tentando recuperação

```
// Try to save the address book.
boolean successful = false;
int attempts = 0;
do {
    try {
        addressbook.saveToFile(filename);
        successful = true;
    }
    catch(IOException e) {
        System.out.println("Unable to save to " + filename);
        attempts++;
        if(attempts < MAX_ATTEMPTS) {
            filename = um nome de arquivo alternativo;
        }
    }
} while(!successful && attempts < MAX_ATTEMPTS);
if(!successful) {
    Informe o problema e desista;
}
```

Evitando uma exceção

```
// Usa o método correto para colocar detalhes
// no catálogo de endereços.
if(book.keyInUse(details.getName() ||
    book.keyInUse(details.getPhone())) {
    book.changeDetails(details);
}
else {
    book.addDetails(details);
}
```

Evitando uma exceção

```
// Usa o método correto para colocar detalhes
// no catálogo de endereços.
if(book.keyInUse(details.getName() ||
    book.keyInUse(details.getPhone())) {
    book.changeDetails(details);
}
else {
    book.addDetails(details);
}
```

- O método `addDetails` poderia lançar uma exceção não verificada.

Definindo novas exceções

- São usadas quando as classes de exceção padrão não descrevem satisfatoriamente a natureza do problema.
- Novas classes mais descritivas podem ser definidas utilizando herança.

Definindo novas exceções

```
public class NoMatchingDetailsException extends Exception
{
    private String key;

    public NoMatchingDetailsException(String key)
    {
        this.key = key;
    }

    public String getKey()
    {
        return key;
    }

    public String toString()
    {
        return "No details matching '" + key +
            "' were found.";
    }
}
```


Outline

- 1 Tipos de erros
- 2 Erro lógico
- 3 Caso: Calculadora
- 4 Exceções
- 5 Assertivas**

Assertivas

- Usadas para verificações de consistência internas.

Assertivas

- Usadas para verificações de consistência internas.
- Usadas durante o desenvolvimento e normalmente removidas na versão de produção.

Assertivas

- Usadas para verificações de consistência internas.
- Usadas durante o desenvolvimento e normalmente removidas na versão de produção.
 - Por exemplo, via uma opção de tempo de compilação.

Tipos

- É expressada de forma booleana

Tipos

- É expressada de forma booleana
- A expressão booleana expressa algo que deve ser verdadeiro nesse ponto.

Tipos

- É expressada de forma booleana
- A expressão booleana expressa algo que deve ser verdadeiro nesse ponto.
- Um `AssertionError` é lançado se a assertiva for falsa.

Tipos

- É expressada de forma booleana
- A expressão booleana expressa algo que deve ser verdadeiro nesse ponto.
- Um `AssertionError` é lançado se a assertiva for falsa.

```
public void removeDetails(String key)
{
    if(key == null){
        throw new IllegalArgumentException("...");
    }
    if(keyInUse(key)) {
        ContactDetails details = book.get(key);
        book.remove(details.getName());
        book.remove(details.getPhone());
        numberOfEntries--;
    }
    assert !keyInUse(key) ;
    assert consistentSize() :
        "Inconsistent book size in removeDetails";
}
```


Tipos

- Não são alternativa para as exceções de lançamento.
- Use para verificações internas.
- Remova do código de produção.
- Não inclui funcionalidade normal:

```
// Uso incorreto:  
assert book.remove(name) != null;
```

Revisão

Erros de runtime surgem por muitas razões.

- Uma chamada de cliente inadequada para um objeto servidor.
- Um servidor incapaz de atender a uma solicitação.
- Erro de programação no cliente e/ou servidor.

Revisão

- Em geral, os erros de runtime resultam em falha de programa.
- A programação defensiva antecipa erros, tanto no cliente como no servidor.
- As exceções fornecem um mecanismo de relatório e recuperação.

Leitura

- Capítulo 12 (Prog. Orient. a Obj. usando Java - 4th Edition)
- Capítulo 11 (Java: Como programar - 8th Edition)