Orientação a Objeto - Tratamento de Erros Usando Exceções

Objetivos da Seção

- Aprender a lidar com condições especiais normais e condições especiais anormais, utilizando o mecanismo de exceção

O Problema: Como Tratar Erros

- Lembre o progama Cartas7.java?
- Ei-lo de novo aqui embaixo para refrescar a memória

```
/*
 * Leitura em laço, com tratamento de erro
 */

import p1.aplic.cartas.*;
import p1.io.*;

public class Cartas7 {
  public static void main(String[] args) {
    int     rodadas = 0;
    String  resposta;
    boolean respostaOK = false;

    while(!respostaOK) {
      resposta = Entrada.in.lerLinha("Quantas rodadas quer jogar?");
      if(resposta == null) {
        // fim de entrada
        System.exit(0);
      }
      rodadas = Integer.parseInt(resposta);
      if(rodadas <= 0 ) {
        System.err.println("Forneca um numero positivo, por favor.");
      } else {
        respostaOK = true;
      }
    }

    // MaiorCarta é um jogo onde quem detém a maior carta ganha a rodada
    new MaiorCarta().joga(rodadas);
  } // main
} // Cartas7
```

- Agora, vamos lembrar a classe MaiorCarta.java

```
/*
 * Desenvolvido para a disciplina Programacao 1
 * Curso de Bacharelado em Ciência da Computação
 * Departamento de Sistemas e Computação
 * Universidade Federal da Paraíba
 *
 * Copyright (C) 1999 Universidade Federal da Paraíba.
 * Não redistribuir sem permissão.
 */

package p1.aplic.cartas;

import java.util.*;

/**
 * Um jogo de cartas simples.
 * Cada jogador recebe uma carta do baralho.
 * A maior carta ganha.
 * Repete para cada rodada.
 *
 * @author Jacques Philippe Sauvé, jacques@dsc.ufpb.br
```

```
 * @version 1.0
 * <br>
 * Copyright (C) 1999 Universidade Federal da Paraíba.
 */
public class MaiorCarta {
  private int     suasVitórias;  // pontuação
  private int     minhasVitórias;
  private Baralho baralho;

  /**
   * Construtor do jogo.
   */
   public MaiorCarta() {
    suasVitórias = 0;
    minhasVitórias = 0;
    // Factory method para permitir jogar com outros baralhos
    // com override de criaBaralho()
    baralho = criaBaralho();
    baralho.baralhar();
  }

  /**
   * Cria um baralho para jogar.
   * @return O baralho.
   */
  protected Baralho criaBaralho() {
    return new Baralho();
  }

  /**
   * Joga o jogo de Maior Carta.
   * @param rodadas O número de rodadas a jogar.
   */
  public void joga(int rodadas) {
    for(int i = 0; i < rodadas; i++) {
      Carta suaCarta = baralho.pegaCarta();
      System.out.print("Sua carta: " + suaCarta + " ");
      Carta minhaCarta = baralho.pegaCarta();
      System.out.print("Minha carta: " + minhaCarta + " ");
      if(suaCarta.compareTo(minhaCarta) > 0) {
        System.out.println("Voce ganha.");
        suasVitórias++;
      } else if(suaCarta.compareTo(minhaCarta) < 0) {
        System.out.println("Eu ganho.");
        minhasVitórias++;
      } else {
        System.out.println("Empate.");
      }
    }
    System.out.println("Voce ganhou " + suasVitórias +
      " vezes, eu ganhei " + minhasVitórias + " vezes, " +
      (rodadas-suasVitórias-minhasVitórias) + " empates.");
  }
}
```

- Vamos rodar o programa Cartas7.java com 30 rodadas
  - A saída segue:

```
Quantas rodadas quer jogar?30
Sua carta: DOIS de PAUS Minha carta: TRES de PAUS Eu ganho.
Sua carta: DOIS de ESPADAS Minha carta: SEIS de ESPADAS Eu ganho.
...
Sua carta: QUATRO de PAUS Minha carta: OITO de OUROS Eu ganho.
Sua carta: CINCO de ESPADAS Minha carta: SETE de ESPADAS Eu ganho.
Sua carta: null Minha carta: null Exception in thread "main" java.lang.NullPointerException
        at p1.aplic.cartas.MaiorCarta.joga(MaiorCarta.java:61)
        at Cartas7.main(Cartas7.java:30)
```

- Observe que o programa pipocou na cara do usuário de forma deselegante
  - A "forma deselegante" aqui se chama NullPointerException
- Onde houve um "null pointer"?

- Baralho.pegaCarta() retornou null
- MaiorCarta se lascou na linha 61, porque suaCarta tem valor null

```
if(suaCarta.compareTo(minhaCarta) > 0) { // linha 61
```

- Como tratar condições de erro deste tipo de forma elegante?
- Em geral, que mecanismo podemos usar para tratar erros?
- É importante diferenciar o descobrimento do erro e o tratamento do erro
  - É muito frequente descobrir algo errado em um lugar mas querer tratar o erro em outro lugar
  - Por exemplo, tratar o erro de baralho vazio em pegaCarta() é ruim porque é um método de "baixo nível" que não sabe sequer que tipo de interface está sendo usada (gráfica, a caractere), etc.
    - Não seria apropriado fazer um println e exit

Uma solução: Exceções

- Vamos usar um mecanismo novo para retornar erros
- O retorno normal de valores por um método usa "return"
- O retorno anormal (indicando erro) usa outra palavra para retornar do método
  - A palavra é throw
- Da mesma forma que "return", "throw" retorna imediatamente do método
- Diferentemente de "return", "throw" só retorna objetos especiais chamados exceções
  - A exceção pode conter uma mensagem indicando o erro que ocorreu
- "throw" faz com que todos os métodos chamados retornem, até o ponto em que algum método captura a exceção para tratar o erro
  - Essa captura é feita com um bloco try-catch
- Um exemplo de captura já foi visto em Cartas12.java (clique aqui)
- Agora, vamos ver como montar esse circo
- Começamos com BaralhoSeguro.java, mas mostramos apenas o método que muda
  - É o ponto onde detectamos um erro e lançamos uma exceção

```
/**
 * Retira uma carta do topo do baralho e a retorna. A carta é removida do baralho.
 * @return A carta retirada do baralho.
 */
public Carta pegaCarta() throws BaralhoVazioException {
  if(númeroDeCartas() == 0) {
    throw new BaralhoVazioException("Baralho esta vazio");
  }
  return (Carta)baralho.remove(númeroDeCartas()-1);
}
```

- Observe a cláusula "throws" que diz que tipo de exceção o método pode lançar
  - É importante dizer tanto o que um método retorna normalmente quanto o que ele retorna quando há erro
- Java tem muitas exceções, para indicar várias condições de erro
  - Aqui, estamos criando nossa própria exceção para representar o erro
  - Existem outras exceções do sistema (exemplo, ArithmeticException se dividir por zero)
- O código de BaralhoVazioException segue (ver BaralhoVazioException.java)

```
package p1.aplic.cartas;

/**
 * Classe de Exceção quando tenta-se poegar uma carta de um baralho vazio.
 *
 * @author   Jacques Philippe Sauvé, jacques@dsc.ufpb.br
 * @version 1.1
```

```
   * <br>
   * Copyright (C) 2001 Universidade Federal da Paraíba.
   */
  public class BaralhoVazioException extends Exception {

    /**
     * Construtor de exceção informando que o baralho está vazio.
     * @param motivo O motivo pelo qual não se pode fechar a conta.
     */
    public BaralhoVazioException(String motivo) {
      super(motivo);
    }
  }
```

- Agora, vamos ver a classe MaiorCartaSegura.java onde capturaremos a exceção e faremos algo a respeito do erro

```
  /*
   * Desenvolvido para a disciplina Programacao 1
   * Curso de Bacharelado em Ciência da Computação
   * Departamento de Sistemas e Computação
   * Universidade Federal da Paraíba
   *
   * Copyright (C) 1999 Universidade Federal da Paraíba.
   * Não redistribuir sem permissão.
   */

  package p1.aplic.cartas;

  import java.util.*;

  /**
   * Como a classe MaiorCarta, mas tratando de baralho vazio com exceção.
   * Não usa herança para que os alunos possam ver o exemplo antes de ver herança.
   *
   * @author Jacques Philippe Sauvé, jacques@dsc.ufpb.br
   * @version 1.1
   * <br>
   * Copyright (C) 1999 Universidade Federal da Paraíba.
   */
  public class MaiorCartaSegura {
    private int     suasVitórias;  // pontuação
    private int     minhasVitórias;
    private int     empates;
    private BaralhoSeguro baralho;

    /**
     * Construtor do jogo.
     */
    public MaiorCartaSegura() {
      suasVitórias = 0;
      minhasVitórias = 0;
      empates = 0;
      // Factory method para permitir jogar com outros baralhos
      // com override de criaBaralho()
      baralho = criaBaralho();
      baralho.baralhar();
    }

    /**
     * Cria um baralho para jogar.
     * @return O baralho.
     */
    protected BaralhoSeguro criaBaralho() {
      return new BaralhoSeguro();
    }

    /**
     * Joga o jogo de Maior Carta.
     * @param rodadas O número de rodadas a jogar.
     */
    public void joga(int rodadas) {
      for(int i = 0; i < rodadas; i++) {
```

```
      Carta suaCarta, minhaCarta;
      try {
        suaCarta = baralho.pegaCarta();
        minhaCarta = baralho.pegaCarta();
      } catch(BaralhoVazioException e) {
        System.out.println(e.getMessage());
        break;
      }
      System.out.print("Sua carta: " + suaCarta + " ");
      System.out.print("Minha carta: " + minhaCarta + " ");
      if(suaCarta.compareTo(minhaCarta) > 0) {
        System.out.println("Voce ganha.");
        suasVitórias++;
      } else if(suaCarta.compareTo(minhaCarta) < 0) {
        System.out.println("Eu ganho.");
        minhasVitórias++;
      } else {
        System.out.println("Empate.");
        empates++;
      }
    }
    System.out.println("Voce ganhou " + suasVitórias +
      " vezes, eu ganhei " + minhasVitórias + " vezes, " +
      empates + " empates.");
  }
}
```

- Vemos acima como capturar uma exceção com try-catch e como obter a mensagem que está dentro da exceção (chamando e.getMessage())
- Claro que só imprimir a mensagem de erro não é suficiente, tem que tratar o erro
  - Aqui, fazemos isso caindo fora do loop com break
- Observe também que o cálculo de empates mudou (com respeito a MaiorCarta) porque é possível que algumas rodadas não sejam jogadas e o cálculo anterior (em MaiorCarta) estaria errado
- O tratamento do erro poderia ser diferente
  - Exemplo: poderia-se tentar continuar o jogo arrumando outro baralho
  - Pergunta: qual seria a forma mais limpa de fazer o jogo funcionar para qualquer número de rodadas?
    - Minha sugestão: criar uma abstração (classe) nova BaralhoInfinito
    - Se este BaralhoInfinito usa exceções ou não para funcionar depende da forma de implementação
    - Acho que eu não usaria exeções para isso, por dois motivos:
      - É fácil implementar BaralhoInfinito sem exceções
      - Exceções devem ser usadas para condições anormais não esperadas, e isso não é o caso aqui
- Agora, podemos jogar com visto em Cartas11.java

```
/*
 * Para uso em exemplo de exceções
 */

import p1.aplic.cartas.*;
import p1.io.*;

public class Cartas11 {
  public static void main(String[] args) {
    int     rodadas = 0;
    String  resposta;
    boolean respostaOK = false;

    while(!respostaOK) {
      resposta = Entrada.in.lerLinha("Quantas rodadas quer jogar?");
      if(resposta == null) {
        // fim de entrada
        System.exit(0);
      }
      rodadas = Integer.parseInt(resposta);
      if(rodadas <= 0 ) {
```

```
          System.err.println("Forneca um numero positivo, por favor.");
        } else {
          respostaOK = true;
        }
      }

      // MaiorCartaSegura é um jogo onde quem detém a maior carta ganha a rodada
      new MaiorCartaSegura().joga(rodadas);
    } // main
} // Cartas11
```

- A saída do programa segue:

```
Quantas rodadas quer jogar?30
Sua carta: DOIS de COPAS Minha carta: DEZ de ESPADAS Eu ganho.
Sua carta: QUATRO de PAUS Minha carta: TRES de OUROS Voce ganha.
...
Sua carta: SEIS de OUROS Minha carta: DAMA de COPAS Eu ganho.
Sua carta: DEZ de OUROS Minha carta: OITO de COPAS Voce ganha.
Baralho esta vazio
Voce ganhou 14 vezes, eu ganhei 11 vezes, 1 empates.
```

- O próximo exemplo é semelhante mas capturando a exceção "um nível acima" na sequência de chamadas de métodos
  - Observe a diferença da saída quando joga 30 rodadas (comparado com Cartas11)
- Aqui está MaiorCartaSegura2.java

```
/*
 * Desenvolvido para a disciplina Programacao 1
 * Curso de Bacharelado em Ciência da Computação
 * Departamento de Sistemas e Computação
 * Universidade Federal da Paraíba
 *
 * Copyright (C) 1999 Universidade Federal da Paraíba.
 * Não redistribuir sem permissão.
 */

package p1.aplic.cartas;

import java.util.*;

/**
 * Como a classe MaiorCartaSegura, mas com tratamento de exceção diferente.
 * Não usa herança para que os alunos possam ver o exemplo antes de ver herança.
 *
 * @author Jacques Philippe Sauvé, jacques@dsc.ufpb.br
 * @version 1.0
 * <br>
 * Copyright (C) 1999 Universidade Federal da Paraíba.
 */
public class MaiorCartaSegura2 {
  private int     suasVitórias;  // pontuação
  private int     minhasVitórias;
  private int     empates;
  private BaralhoSeguro baralho;

  /**
   * Construtor do jogo.
   */
  public MaiorCartaSegura2() {
    suasVitórias = 0;
    minhasVitórias = 0;
    empates = 0;
    // Factory method para permitir jogar com outros baralhos
    // com override de criaBaralho()
    baralho = criaBaralho();
    baralho.baralhar();
  }

  /**
```

```
     * Cria um baralho para jogar.
     * @return O baralho.
     */
    protected BaralhoSeguro criaBaralho() {
      return new BaralhoSeguro();
    }

    /**
     * Joga o jogo de Maior Carta.
     * @param rodadas O número de rodadas a jogar.
     */
    public void joga(int rodadas) throws BaralhoVazioException {
      for(int i = 0; i < rodadas; i++) {
        Carta suaCarta, minhaCarta;
        suaCarta = baralho.pegaCarta();
        minhaCarta = baralho.pegaCarta();
        System.out.print("Sua carta: " + suaCarta + " ");
        System.out.print("Minha carta: " + minhaCarta + " ");
        if(suaCarta.compareTo(minhaCarta) > 0) {
          System.out.println("Voce ganha.");
          suasVitórias++;
        } else if(suaCarta.compareTo(minhaCarta) < 0) {
          System.out.println("Eu ganho.");
          minhasVitórias++;
        } else {
          System.out.println("Empate.");
          empates++;
        }
      }
      System.out.println("Voce ganhou " + suasVitórias +
                         " vezes, eu ganhei " + minhasVitórias + " vezes, " +
                         empates + " empates.");
    }
  }
```

- Observe que o método joga() não captura a exceção mas a "deixa passar"
- Aqui está Cartas12.java

```
  /*
   * Para uso em exemplo de exceções
   */

  import p1.aplic.cartas.*;
  import p1.io.*;

  public class Cartas12 {
    public static void main(String[] args) {
      int     rodadas = 0;
      String  resposta;
      boolean respostaOK = false;

      while(!respostaOK) {
        resposta = Entrada.in.lerLinha("Quantas rodadas quer jogar?");
        if(resposta == null) {
            // fim de entrada
            System.exit(0);
        }
        rodadas = Integer.parseInt(resposta);
        if(rodadas <= 0 ) {
          System.err.println("Forneca um numero positivo, por favor.");
        } else {
          respostaOK = true;
        }
      }

      // MaiorCartaSegura é um jogo onde quem detém a maior carta ganha a rodada
      try {
        new MaiorCartaSegura2().joga(rodadas);
      } catch(Exception e) {
        System.out.println(e.getMessage());
      }
    } // main
  } // Cartas12
```

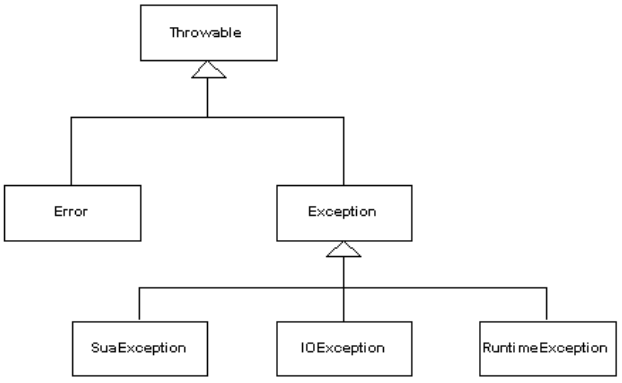
- Aqui está a saída, diferente do jogo Cartas11.java

```
Quantas rodadas quer jogar?30
Sua carta: OITO de PAUS Minha carta: OITO de COPAS Empate.
Sua carta: NOVE de OUROS Minha carta: DEZ de COPAS Eu ganho.
...
Sua carta: QUATRO de OUROS Minha carta: CINCO de OUROS Eu ganho.
Sua carta: DAMA de PAUS Minha carta: AS de COPAS Voce ganha.
Baralho esta vazio
```

- Observe que capturamos Exception e não BaralhoVazioException
  - Funciona porque um BaralhoVazioException "é um" Exception devido à herança
  - Poderíamos ter capturado BaralhoVazioException também

Detalhes finais

- A hierarquia de exceções em Java é parecida com o que segue:



- Error é lançada quando há um erro interno do Java (é raro)
- RuntimeException (NullPointerException, ...) é lançado quando seu programa tem um bug que você não tratou
- Error e RuntimeException são "unchecked"
- O resto é "checked" e você é obrigado a declarar no retorno do método e tratar
- Podemos incluir mais informação num exceção além da mensagem
- Veja como exemplo NaoPodeFecharContaException.java

```
package p1.aplic.banco;

/**
 * Classe de Exceção quando tenta-se fechar uma conta sem estar com saldo zerado.
 *
 * @author   Jacques Philippe Sauvé, jacques@dsc.ufpb.br
 * @version 1.1
 * <br>
 * Copyright (C) 1999 Universidade Federal da Paraíba.
 */
public class NaoPodeFecharContaException extends Exception {
  Conta conta;

  /**
   * Construtor de exceção informando que uma conta não pode ser fechada.
   * @param conta A conta que não pode ser fechada.
   * @param motivo O motivo pelo qual não se pode fechar a conta.
   */
  public NaoPodeFecharContaException(Conta conta, String motivo) {
    super(motivo);
    this.conta = conta;
  }
}
```

- Podemos usar a exceção acima como segue

```
package p1.aplic.banco;

import java.util.*;
import java.io.*;
import p1.aplic.geral.*;

/**
 * Classe abstrata de conta bancária com implementações default de alguns métodos.
 *
 * @author   Jacques Philippe Sauvé, jacques@dsc.ufpb.br
 * @version 1.1
 * <br>
 * Copyright (C) 1999 Universidade Federal da Paraíba.
 */
public abstract class Conta implements Serializable {
  // ...
  /**
   * Fecha a conta.
   * @throws NaoPodeFecharContaException Quando se tenta fechar uma conta com saldo não zero.
   */
  public void fechar() throws NaoPodeFecharContaException  {
    Agencia.fecharConta(número);
  }
}
```

- Aqui está Agencia.fecharConta

```
  /**
   * Fecha uma conta.
   * @param número O número da conta a fechar.
   * @throws NaoPodeFecharContaException se a conta não existir ou tiver saldo
   */
  static void fecharConta(int número) throws NaoPodeFecharContaException {
    abrirCaixa();
    Conta c = localizarConta(número);
    if(c == null) {
      throw new NaoPodeFecharContaException(c, "Conta nao existe");
    }
    if(c.getSaldo() != 0.0) {
      throw new NaoPodeFecharContaException(c, "Saldo nao esta zerado");
    }
    contas.remove(Integer.toString(número));
}
```

- É possível que um trecho de código posssa lançar mais de uma exceção e podemos capturar e tratar todas elas

```
  try {
    // trecho que pode lançar várias exceções
  } catch(NaoPodeFecharContaException e) {
    // trata NaoPodeFecharContaException
  } catch(IOException e) {
    // trata IOException
  } catch(Exception e) {
    // trata Exception não capturado acima
  }
```

- Tem vários outros assuntos relacionados com exceções que você deveria estudar nos livros de Java
  - Relançando exceções
  - Quando usar exceções e quando usar testes normais
  - Como usar a palavra chave "finally" de Java
  - Quais são as RuntimeException comuns que Java lança
  - Quais são as exceções na API Java

oo-6 programa anterior próxima