

Programação Orientada a Objetos

Prof. Paulo André Castro
pauloac@ita.br
www.comp.ita.br/~pauloac
ITA – Stefanini

Paulo André Castro POO 1 ITA – Stefanini

Planejamento

- Aula 4
 - Programando Interfaces Gráficas com Java - II
- **Aula 5**
 - **Tipos Genéricos**
 - **Conexão com outros programas em Rede**
- Aula 6
 - Conectividade com Banco de Dados (JDBC)
 - Padrão de projeto para acesso a Dados: DAO Design Pattern
- Aula 7
 - Introdução a Servlets e JSP
- Aula 8
 - XML
 - Introdução a Web Services

Paulo André Castro POO 2 ITA – Stefanini

5.1. Tipos Genéricos

Sumário

Introdução
Métodos Genéricos
Classes Parametrizadas (Genéricas)
Utilização de wildcards

Paulo André Castro POO 3 ITA – Stefanini

Introdução – Métodos Sobrecarregados

// Utilizando métodos sobrecarregados para imprimir um array de diferentes tipos.

```
public class OverloadedMethods
{
    // método printArray para imprimir um array de Integer
    public static void printArray( Integer[] inputArray )
    {
        // exibe elementos do array
        for ( Integer element : inputArray )
            System.out.printf( "%s ", element );

        System.out.println();
    } // fim do método printArray
}
```

Paulo André Castro POO 4 ITA – Stefanini

Introdução – Métodos Sobrecarregados 2

```
// método printArray para imprimir um array de Double
public static void printArray( Double[] inputArray )
{
    // exibe elementos do array
    for ( Double element : inputArray )
        System.out.printf( "%s ", element );
    System.out.println();
} // fim do método printArray

// método printArray para imprimir um array de Character
public static void printArray( Character[] inputArray )
{
    // exibe elementos do array
    for ( Character element : inputArray )
        System.out.printf( "%s ", element );

    System.out.println();
} // fim do método printArray
```

Paulo André Castro POO 5 ITA – Stefanini

Introdução – Métodos Sobrecarregados - 3

```
public static void main( String args[] )
{
    // cria arrays de Integer, Double e Character
    Integer[] integerArray = { 1, 2, 3, 4, 5, 6 };
    Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };
    Character[] characterArray = { 'H', 'E', 'L', 'L', 'O' };

    System.out.println( "Array integerArray contains:" );
    printArray( integerArray ); // passa um array de Integers
    System.out.println( "\nArray doubleArray contains:" );
    printArray( doubleArray ); // passa um array Doubles
    System.out.println( "\nArray characterArray contains:" );
    printArray( characterArray ); // passa um array de Characters
} // fim de main
} // fim da classe OverloadedMethods
```

Paulo André Castro POO 6 ITA – Stefanini

Resultado – Métodos Sobrecarregados

Array integerArray contains:

1 2 3 4 5 6

Array doubleArray contains:

1.1 2.2 3.3 4.4 5.5 6.6 7.7

Array characterArray contains:

H E L L O

Método Genérico

// Utilizando métodos genéricos para imprimir diferentes tipos de arrays.

```
public class GenericMethodTest
{
    // método genérico printArray
    public static < E > void printArray( E[] inputArray )
    {
        // exibe elementos do array
        for ( E element : inputArray )
            System.out.printf( "%s ", element );

        System.out.println();
    } // fim do método printArray
}
```

Método Genérico - 2

```
public static void main( String args[] )
{
    // cria arrays de Integer, Double e Character
    Integer[] integerArray = { 1, 2, 3, 4, 5, 6 };
    Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };
    Character[] characterArray = { 'H', 'E', 'L', 'L', 'O' };
    System.out.println( "Array integerArray contains:" );
    printArray( integerArray ); // passa um array de Integers
    System.out.println( "\nArray doubleArray contains:" );
    printArray( doubleArray ); // passa um array Doubles
    System.out.println( "\nArray characterArray contains:" );
    printArray( characterArray ); // passa um array de Characters
} // fim de main
} // fim da classe GenericMethodTest
```

Resultado – Método Genérico

Array integerArray contains:

1 2 3 4 5 6

Array doubleArray contains:

1.1 2.2 3.3 4.4 5.5 6.6 7.7

Array characterArray contains:

H E L L O

Exemplo 2 de Método Genérico

```
public class MaximumTest{
    // determina o maior dos três objetos Comparable
    public static < T extends Comparable< T > > T maximum( T x, T y, T z ) {
        T max = x; // supõe que x é inicialmente o maior

        if ( y.compareTo( max ) > 0 )
            max = y; // y é o maior até agora

        if ( z.compareTo( max ) > 0 )
            max = z; // z é o maior

        return max; // retorna o maior objeto
    } // fim do método Maximum
}
```

Exemplo 2 de Método Genérico

```
public static void main( String args[] )
{
    System.out.printf( "Maximum of %d, %d and %d is %d\n\n", 3, 4, 5,
        maximum( 3, 4, 5 ) );
    System.out.printf( "Maximum of %.1f, %.1f and %.1f is %.1f\n\n",
        6.6, 8.8, 7.7, maximum( 6.6, 8.8, 7.7 ) );
    System.out.printf( "Maximum of %s, %s and %s is %s\n", "pear",
        "apple", "orange", maximum( "pear", "apple", "orange" ) );
} // fim de main
} // fim da classe MaximumTest
```

Resultado – Exemplo 2 de Método Genérico

Maximum of 3, 4 and 5 is 5

Maximum of 6,6, 8,8 and 7,7 is 8,8

Maximum of pear, apple and orange is pear

Paulo André Castro

POO

ITA – Stefanini 13

Classe Genérica

- Classe Genérica
 - Criando uma classe que trabalha com vários tipos de classes

Paulo André Castro

POO

ITA – Stefanini 14

Exemplo Classe Genérica – Pilha

```
public class Stack<E> {
    private final int size; // número de elementos na pilha
    private int top; // localização do elemento superior
    private E[] elements; // array que armazena elementos na pilha

    // construtor sem argumento cria uma pilha do tamanho padrão
    public Stack() {
        this( 10 ); // tamanho padrão da pilha
    } // fim do construtor sem argumentos da classe Stack

    // construtor cria uma pilha com o número especificado de elementos
    public Stack( int s ) {
        size = s > 0 ? s : 10; // configura o tamanho da Stack
        top = -1; // Stack inicialmente vazia

        elements = ( E[] ) new Object[ size ]; // cria o array
    } // fim do construtor de Stack
}
```

Paulo André Castro

POO

ITA – Stefanini 15

Exemplo Classe Genérica – Pilha - 2

```
// insere o elemento na pilha; se bem-sucedido retorna true;
// caso contrário, lança uma FullStackException
public void push( E pushValue ) {
    if ( top == size - 1 ) // se a pilha estiver cheia
        throw new FullStackException( String.format(
            "Stack is full, cannot push %s", pushValue ) );

    elements[ ++top ] = pushValue; // insere pushValue na Stack
} // fim do método push

// retorna o elemento superior se não estiver vazia; do contrário lança uma
// EmptyStackException
public E pop() {
    if ( top == -1 ) // se pilha estiver vazia
        throw new EmptyStackException( "Stack is empty, cannot pop" );

    return elements[ top-- ]; // remove e retorna o elemento superior da Stack
} // fim do método pop
} // fim da classe Stack <E>
```

Paulo André Castro

POO

ITA – Stefanini 16

Exemplo Classe Genérica – Pilha – StackTest - 1/6

```
public class StackTest {
    private double[] doubleElements = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6 };
    private int[] integerElements = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };

    private Stack< Double > doubleStack; // a pilha armazena objetos Double
    private Stack< Integer > integerStack; // a pilha armazena objetos Integer

    // testa objetos Stack
    public void testStacks()
    {
        doubleStack = new Stack< Double >( 5 ); // Stack de Doubles
        integerStack = new Stack< Integer >( 10 ); // Stack de Integers

        testPushDouble(); // insere doubles em doubleStack
        testPopDouble(); // remove de doubleStack
        testPushInteger(); // insere ints em intStack
        testPopInteger(); // remove de intStack
    } // fim do método testStacks
}
```

Paulo André Castro

POO

ITA – Stefanini 17

Exemplo Classe Genérica – Pilha – StackTest – 2/6

```
// testa o método push com a pilha de doubles
public void testPushDouble() {
    // insere elementos na pilha
    try {
        System.out.println( "\nPushing elements onto doubleStack" );
        // insere elementos na Stack
        for ( double element : doubleElements ) {
            System.out.printf( "%.1f ", element );
            doubleStack.push( element ); // insere em doubleStack
        } // fim do for
    } // fim do try
    catch ( FullStackException fullStackException ) {
        System.err.println();
        fullStackException.printStackTrace();
    } // fim da captura de FullStackException
} // fim do método testPushDouble
```

Paulo André Castro

POO

ITA – Stefanini 18

Exemplo Classe Genérica – Pilha – StackTest – 3/6

```
// testa o método pop com a pilha de doubles
public void testPopDouble() {
    // Retira elementos da pilha
    try {
        System.out.println( "\nPopping elements from doubleStack" );
        double popValue; // armazena o elemento removido da pilha
        // remove todos os elementos da Stack
        while ( true ) {
            popValue = doubleStack.pop(); // remove de doubleStack
            System.out.printf( "%.1f ", popValue );
        } // fim do while
    } // fim do try
    catch( EmptyStackException emptyStackException ) {
        System.err.println(); emptyStackException.printStackTrace();
    } // fim da captura de EmptyStackException
} // fim do método testPopDouble
```

Paulo André Castro

POO

ITA – Stefanini 19

Exemplo Classe Genérica – Pilha - StackTest – 4/6

```
// testa o método pop com a pilha de integers
public void testPopInteger() {
    // remove elementos da pilha
    try {
        System.out.println( "\nPopping elements from integerStack" );
        int popValue; // armazena o elemento removido da pilha
        // remove todos os elementos da Stack
        while ( true ) {
            popValue = integerStack.pop(); // remove de integerStack
            System.out.printf( "%d ", popValue );
        } // fim do while
    } // fim do try
    catch( EmptyStackException emptyStackException ) {
        System.err.println(); emptyStackException.printStackTrace();
    } // fim da captura de EmptyStackException
} // fim do método testPopInteger
```

Paulo André Castro

POO

ITA – Stefanini 20

Exemplo Classe Genérica – Pilha - StackTest – 5/6

```
// testa o método push com a pilha de integers
public void testPushInteger() {
    // insere elementos na pilha
    try {
        System.out.println( "\nPushing elements onto integerStack" );
        // insere elementos na Stack
        for ( int element : integerElements ) {
            System.out.printf( "%d ", element );
            integerStack.push( element ); // insere em integerStack
        } // fim do for
    } // fim do try
    catch ( FullStackException fullStackException ) {
        System.err.println(); fullStackException.printStackTrace();
    } // fim da captura de FullStackException
} // fim do método testPushInteger
```

Paulo André Castro

POO

ITA – Stefanini 21

Exemplo Classe Genérica – Pilha - StackTest – 6/6

```
public static void main( String args[] )
{
    StackTest application = new StackTest();
    application.testStacks();
} // fim de main
} // fim da classe StackTest
```

Paulo André Castro

POO

ITA – Stefanini 22

Exemplo Classe Genérica – Pilha - FullStackException

```
public class FullStackException extends RuntimeException {
    // construtor sem argumento
    public FullStackException() {
        this( "Stack is full" );
    } // fim do construtor sem argumentos de FullStackException

    // construtor de um argumento
    public FullStackException( String exception ) {
        super( exception );
    } // fim do construtor de FullStackException de um argumento
} // fim da classe FullStackException
```

Paulo André Castro

POO

ITA – Stefanini 23

Exemplo Classe Genérica – Pilha - EmptyStackException

```
public class EmptyStackException extends RuntimeException
{
    // construtor sem argumento
    public EmptyStackException()
    {
        this( "Stack is empty" );
    } // fim do construtor sem argumentos de EmptyStackException

    // construtor de um argumento
    public EmptyStackException( String exception )
    {
        super( exception );
    } // fim do construtor de um argumento de EmptyStackException
} // fim da classe EmptyStackException
```

Paulo André Castro

POO

ITA – Stefanini 24

Resultado – Classe genérica Stack – 1/2

Pushing elements onto doubleStack

1,1 2,2

StackExample.FullStackException: Stack is full, cannot push 6,6

at StackExample.Stack.push(Stack.java:31)

at StackExample.StackTest.testPushDouble(StackTest.java:37)

at StackExample.StackTest.testStacks(StackTest.java:19)

at StackExample.StackTest.main(StackTest.java:118)

StackExample.EmptyStackException: Stack is empty, cannot pop

at StackExample.Stack.pop(Stack.java:41)

at StackExample.StackTest.testPopDouble(StackTest.java:59)

at StackExample.StackTest.testStacks(StackTest.java:20)

at StackExample.StackTest.main(StackTest.java:118)

Paulo André Castro

POO

ITA – Stefanini 26

Resultado – Classe genérica Stack – 2/2

StackExample.FullStackException: Stack is full, cannot push 11

at StackExample.Stack.push(Stack.java:31)

at StackExample.StackTest.testPushInteger(StackTest.java:82)

at StackExample.StackTest.testStacks(StackTest.java:21)3,3 4,4 5,5 6,6

Popping elements from doubleStack

5,5 4,4 3,3 2,2 1,1

Pushing elements onto integerStack

1 2 3 4 5 6 7 8 9 10 11

at StackExample.StackTest.main(StackTest.java:118)

StackExample.EmptyStackException: Stack is empty, cannot pop

at StackExample.Stack.pop(Stack.java:41)

at StackExample.StackTest.testPopInteger(StackTest.java:104)

at StackExample.StackTest.testStacks(StackTest.java:22)

at StackExample.StackTest.main(StackTest.java:118)

Popping elements from integerStack

10 9 8 7 6 5 4 3 2 1

Paulo André Castro

POO

ITA – Stefanini 26

Métodos Genéricos e Classes Genéricas

- StackTest tem métodos praticamente idênticos: testPushInteger, testPushDouble, testPopInteger, testPopDouble
- Utilizando Métodos Genéricos para testar uma classe genérica

Paulo André Castro

POO

ITA – Stefanini 27

Métodos e Classes Genéricas – StackTest2

```
public class StackTest2 {
    private Double [] doubleElements = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6 };
    private Integer [] integerElements = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
    private Stack< Double > doubleStack; // a pilha armazena objetos Double
    private Stack< Integer > integerStack; // a pilha armazena objetos Integer

    // testa objetos Stack
    public void testStacks() {
        doubleStack = new Stack< Double >( 5 ); // Stack de Doubles
        integerStack = new Stack< Integer >( 10 ); // Stack de Integers
        testPush( "doubleStack", doubleStack, doubleElements );
        testPop( "doubleStack", doubleStack );
        testPush( "integerStack", integerStack, integerElements );
        testPop( "integerStack", integerStack );
    } // fim do método testStacks
}
```

Paulo André Castro

POO

ITA – Stefanini 28

Métodos e Classes Genéricas – StackTest2 - testPush

```
// método genérico testPush insere elementos em uma Stack
public < T > void testPush( String name, Stack< T > stack, T[] elements ) {
    // insere elementos na pilha
    try {
        System.out.printf( "\nPushing elements onto %s\n", name );
        // insere elementos na Stack
        for ( T element : elements ) {
            System.out.printf( "%s ", element );
            stack.push( element ); // insere o elemento na pilha
        }
    } // fim do try
    catch ( FullStackException fullStackException ) {
        System.out.println(); fullStackException.printStackTrace();
    } // fim da captura de FullStackException
} // fim do método testPush
```

Paulo André Castro

POO

ITA – Stefanini 29

Métodos e Classes Genéricas – StackTest2 - testPop

```
// método genérico testPop remove elementos de uma Stack
public < T > void testPop( String name, Stack< T > stack ) {
    // remove elementos da pilha
    try {
        System.out.printf( "\nPopping elements from %s\n", name );
        T popValue; // armazena o elemento removido da pilha
        // remove elementos da Stack
        while ( true ) {
            popValue = stack.pop(); // remove da pilha
            System.out.printf( "%s ", popValue );
        } // fim do while
    } // fim do try
    catch ( EmptyStackException emptyStackException ) {
        System.out.println(); emptyStackException.printStackTrace();
    } // fim da captura de EmptyStackException
} // fim do método testPop
```

Paulo André Castro

POO

ITA – Stefanini 30

Métodos e Classes Genéricas – StackTest2 – main

```
public static void main( String args[] )
{
    StackTest2 application = new StackTest2();
    application.testStacks();
} // fim de main
} // fim da classe StackTest2
```

Paulo André Castro

POO

ITA – Stefanini 31

Resultado – StackTest2 – 1/2

```
Stack2Example.FullStackException: Stack is full, cannot push 6.6
at Stack2Example.Stack.push(Stack.java:31)
at Stack2Example.StackTest2.testPush(StackTest2.java:39)
at Stack2Example.StackTest2.testStacks(StackTest2.java:20)
at Stack2Example.StackTest2.main(StackTest2.java:75)
Stack2Example.EmptyStackException: Stack is empty, cannot pop
at Stack2Example.Stack.pop(Stack.java:41)
at Stack2Example.StackTest2.testPop(StackTest2.java:61)
at Stack2Example.StackTest2.testStacks(StackTest2.java:21)
at Stack2Example.StackTest2.main(StackTest2.java:75)
Stack2Example.FullStackException: Stack is full, cannot push 11
at Stack2Example.Stack.push(Stack.java:31)
at Stack2Example.StackTest2.testPush(StackTest2.java:39)
at Stack2Example.StackTest2.testStacks(StackTest2.java:22)
at Stack2Example.StackTest2.main(StackTest2.java:75)
```

Paulo André Castro

POO

ITA – Stefanini 32

Resultado – StackTest2 – 2/2

Pushing elements onto doubleStack
1.1 2.2 3.3 4.4 5.5 6.6

Popping elements from doubleStack
5.5 4.4 3.3 2.2 1.1

Pushing elements onto integerStack
1 2 3 4 5 6 7 8 9 10 11

Popping elements from integerStack
10 9 8 7 6 5 4 3 2 1

```
Stack2Example.EmptyStackException: Stack is empty, cannot pop
at Stack2Example.Stack.pop(Stack.java:41)
at Stack2Example.StackTest2.testPop(StackTest2.java:61)
at Stack2Example.StackTest2.testStacks(StackTest2.java:23)
at Stack2Example.StackTest2.main(StackTest2.java:75)
```

Paulo André Castro

POO

ITA – Stefanini 33

Tipos Brutos “Raw Types”

- Tipos brutos “raw types” são classes parametrizadas que são instanciadas sem parâmetros. Ex.:
 - Stack s=new Stack(10);
- Nesse caso, o parâmetro omitido é assumido como Object.
- É possível fazer referências de tipos não brutos para brutos e vice-versa, porém não é recomendável e geram avisos do compilador
 - Stack tipoBruto=new Stack<String>(10);
 - Stack<Integer> tipoEspecifico=new Stack(20);

Paulo André Castro

POO

ITA – Stefanini 34

Exemplo - Tipos “Brutos” – 1/5

```
public class RawTypeTest {
    private Double[] doubleElements = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6 };
    private Integer[] integerElements =
        { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
    // método para testar classes Stack com tipos brutos
    public void testStacks() {
        // Pilha de tipos brutos atribuídos à classe Stack da variável de tipos brutos
        Stack rawTypeStack1 = new Stack( 5 );
        // Stack< Double > atribuído a Stack da variável de tipos brutos
        Stack rawTypeStack2 = new Stack< Double >( 5 );
        // Pilha de tipos crus atribuídos à variável Stack< Integer >
        Stack< Integer > integerStack = new Stack( 10 );
```

Paulo André Castro

POO

ITA – Stefanini 35

Exemplo - Tipos “Brutos” – 2/5

```
//continuação método testStacks
    testPush( "rawTypeStack1", rawTypeStack1, doubleElements );
    testPop( "rawTypeStack1", rawTypeStack1 );
    testPush( "rawTypeStack2", rawTypeStack2, doubleElements );
    testPop( "rawTypeStack2", rawTypeStack2 );
    testPush( "integerStack", integerStack, integerElements );
    testPop( "integerStack", integerStack );
} // fim do método testStacks
```

Paulo André Castro

POO

ITA – Stefanini 36

Exemplo - Tipos "Brutos" – 3/5

```
// método genérico insere elementos na pilha
public < T > void testPush( String name, Stack< T > stack,
    T[] elements ) {
    // insere elementos na pilha
    try {
        System.out.printf( "\nPushing elements onto %s\n", name );
        // insere elementos na Stack
        for ( T element : elements ) {
            System.out.printf( "%s ", element );
            stack.push( element ); // insere o elemento na pilha
        } // fim do for
    } // fim do try
    catch ( FullStackException fullStackException ) {
        System.out.println();    fullStackException.printStackTrace();
    } // fim da captura de FullStackException
} // fim do método testPush
```

Paulo André Castro

POO

ITA – Stefanini 37

Exemplo - Tipos "Brutos" – 4/5

```
// método genérico testPop remove elementos da pilha
public < T > void testPop( String name, Stack< T > stack ) {
    // remove elementos da pilha
    try {
        System.out.printf( "\nPopping elements from %s\n", name );
        T popValue; // armazena o elemento removido da pilha
        // remove elementos da Stack
        while ( true ) {
            popValue = stack.pop(); // remove da pilha
            System.out.printf( "%s ", popValue );
        } // fim do while
    } // fim do try
    catch( EmptyStackException emptyStackException ) {
        System.out.println();    emptyStackException.printStackTrace();
    } // fim da captura de EmptyStackException
} // fim do método testPop
```

Paulo André Castro

POO

ITA – Stefanini 38

Exemplo - Tipos "Brutos" – 5/5

```
public static void main( String args[] )
{
    RawTypeTest application = new RawTypeTest();
    application.testStacks();
} // fim de main
} // fim da classe RawTypeTest
```

Paulo André Castro

POO

ITA – Stefanini 39

Resultado – Tipos "Brutos" – 1/3

Pushing elements onto rawTypeStack1
1.1 2.2 3.3 4.4 5.5 6.6

Popping elements from rawTypeStack1
5.5 4.4 3.3 2.2 1.1 RawTypeExample.FullStackException: Stack is full, cannot push 6.6
at RawTypeExample.Stack.push(Stack.java:31)
at RawTypeExample.RawTypeTest.testPush(RawTypeTest.java:44)
at RawTypeExample.RawTypeTest.testStacks(RawTypeTest.java:23)
at RawTypeExample.RawTypeTest.main(RawTypeTest.java:80)

Pushing elements onto rawTypeStack2
1.1 2.2 3.3 4.4 5.5 6.6

Paulo André Castro

POO

ITA – Stefanini 40

Resultado – Tipos "Brutos" – 2/3

Popping elements from rawTypeStack2
5.5 4.4 3.3 2.2 1.1

RawTypeExample.EmptyStackException: Stack is empty, cannot pop
at RawTypeExample.Stack.pop(Stack.java:41)
at RawTypeExample.RawTypeTest.testPop(RawTypeTest.java:66)
at RawTypeExample.RawTypeTest.testStacks(RawTypeTest.java:24)
at RawTypeExample.RawTypeTest.main(RawTypeTest.java:80)
RawTypeExample.FullStackException: Stack is full, cannot push 6.6
at RawTypeExample.Stack.push(Stack.java:31)
at RawTypeExample.RawTypeTest.testPush(RawTypeTest.java:44)
at RawTypeExample.RawTypeTest.testStacks(RawTypeTest.java:25)
at RawTypeExample.RawTypeTest.main(RawTypeTest.java:80)
RawTypeExample.EmptyStackException: Stack is empty, cannot pop
at RawTypeExample.Stack.pop(Stack.java:41)
at RawTypeExample.RawTypeTest.testPop(RawTypeTest.java:66)
at RawTypeExample.RawTypeTest.testStacks(RawTypeTest.java:26)
at RawTypeExample.RawTypeTest.main(RawTypeTest.java:80)

Paulo André Castro

POO

ITA – Stefanini 41

Resultado – Tipos "Brutos" – 3/3

Pushing elements onto integerStack
1 2 3 4 5 6 7 8 9 10 11

Popping elements from integerStack
10 9 8 7 6 5 4 3 2 1

RawTypeExample.FullStackException: Stack is full, cannot push 11
at RawTypeExample.Stack.push(Stack.java:31)
at RawTypeExample.RawTypeTest.testPush(RawTypeTest.java:44)
at RawTypeExample.RawTypeTest.testStacks(RawTypeTest.java:27)
at RawTypeExample.RawTypeTest.main(RawTypeTest.java:80)
RawTypeExample.EmptyStackException: Stack is empty, cannot pop
at RawTypeExample.Stack.pop(Stack.java:41)
at RawTypeExample.RawTypeTest.testPop(RawTypeTest.java:66)
at RawTypeExample.RawTypeTest.testStacks(RawTypeTest.java:28)
at RawTypeExample.RawTypeTest.main(RawTypeTest.java:80)

Paulo André Castro

POO

ITA – Stefanini 42

Superclasses e subclasses genéricas

- Classes genéricas são invariantes, isto é: Dadas duas classes A e B, List<A> e List nunca serão subtipo ou supertipo um do outro, mesmo que A e B sejam superclasse e subclasse.
- Em arrays, observa-se covariância. Veja o exemplo abaixo:



Paulo André Castro

POO

ITA - Stefanini 43

Superclasses e subclasses genéricas - 2

- A solução para o problema é utilizar Wildcards (caracteres coringas)
- Vejamos em seguida dois exemplos:
 - Um uso correto de método de classe sem wildcard e
 - Um exemplo de método que usa classe genérica sem uso de Wildcard com Erros devido ao subtipo não reconhecido

Paulo André Castro

POO

ITA - Stefanini 44

Exemplo de método que usa classe genérica sem uso de Wildcard

```
public class TotalNumbers {
    public static void main( String[] args ) {
        // create, initialize and output ArrayList of Integers
        // then display total of the elements
        Number[] numbers = { 1, 2.4, 3, 4.1 };
        ArrayList< Number> numberList = new ArrayList<Number >();
        for (Number element: numbers )
            numberList.add( element ); // place each number in integerList
        System.out.printf( "numberList contains: %s\n", integerList );
        System.out.printf( "Total of the elements in numberList: %.1f\n",
            sum( numberList ) );
    } // end main
}
```

Paulo André Castro

POO

ITA - Stefanini 45

Exemplo de método que usa classe genérica sem uso de Wildcard

```
// calculate total of ArrayList elements
public static double sum( ArrayList< Number > list )
{
    double total = 0; // initialize total

    // calculate sum
    for ( Number element : list )
        total += element.doubleValue();

    return total;
} // end method sum
} // end class TotalNumbersErrors
```

Paulo André Castro

POO

ITA - Stefanini 46

Resultado - Uso de método que usa classe genérica sem uso de Wildcard

numberList contains: [1, 2.4, 3, 4.1]
Total of the elements in numberList: 10,5

Paulo André Castro

POO

ITA - Stefanini 47

Exemplo de método que usa classe genérica sem uso de Wildcard com Erros devido ao subtipo não reconhecido

```
public class TotalNumbersErrors {
    public static void main( String[] args ) {
        // create, initialize and output ArrayList of Integers
        // then display total of the elements
        Integer[] integers = { 1, 2, 3, 4 };
        ArrayList< Integer > integerList = new ArrayList< Integer >();
        for ( Integer element : integers )
            integerList.add( element ); // place each number in integerList

        System.out.printf( "integerList contains: %s\n", integerList );
        System.out.printf( "Total of the elements in integerList: %.1f\n",
            sum( integerList ) );
    } // end main
}
```

Paulo André Castro

POO

ITA - Stefanini 48

Exemplo de método que usa classe genérica sem uso de Wildcard com Erros devido ao subtipo não reconhecido

```
// calculate total of ArrayList elements
public static double sum( ArrayList< Number > list )
{
    double total = 0; // initialize total

    // calculate sum
    for ( Number element : list )
        total += element.doubleValue();

    return total;
} // end method sum
} // end class TotalNumbersErrors
```

Paulo André Castro

POO

ITA - Stefanini 49

Classe que utiliza Wildcard para evitar erros de reconhecimento de subtipo

```
public class WildcardTest {
    public static void main( String args[] ) {
        // cria, inicializa e gera saída de ArrayList de Integers, então
        // exibe o total dos elementos
        Integer[] integers = { 1, 2, 3, 4, 5 };
        ArrayList< Integer > integerList = new ArrayList< Integer >();

        // insere elementos na integerList
        for ( Integer element : integers )
            integerList.add( element );

        System.out.printf( "integerList contains: %s\n", integerList );
        System.out.printf( "Total of the elements in integerList: %.0f\n",
            sum( integerList ) );
    }
}
```

Paulo André Castro

POO

ITA - Stefanini 50

Classe que utiliza Wildcard para evitar erros de reconhecimento de subtipo - 2

```
// cria, inicializa e gera saída do ArrayList de Doubles, então
// exibe o total dos elementos
Double[] doubles = { 1.1, 3.3, 5.5 };
ArrayList< Double > doubleList = new ArrayList< Double >();
// insere elementos na doubleList
for ( Double element : doubles )
    doubleList.add( element );

System.out.printf( "doubleList contains: %s\n", doubleList );
System.out.printf( "Total of the elements in doubleList: %.1f\n",
    sum( doubleList ) );
```

```
// cria, inicializa e gera saída de ArrayList de números contendo
// Integers e Doubles e então exibe o total dos elementos
Number[] numbers = { 1, 2.4, 3, 4.1 }; // Integers and Doubles
ArrayList< Number > numberList = new ArrayList< Number >();
```

Paulo André Castro

POO

ITA - Stefanini 51

Classe que utiliza Wildcard para evitar erros de reconhecimento de subtipo - 3

```
// insere elementos na numberList
for ( Number element : numbers )
    numberList.add( element );
System.out.printf( "numberList contains: %s\n", numberList );
System.out.printf( "Total of the elements in numberList: %.1f\n",
    sum( numberList ) );
} // fim de main
// calcula o total de elementos na pilha
public static double sum( ArrayList< ? extends Number > list ) {
    double total = 0; // inicializa o total
    // calcula a soma
    for ( Number element : list )
        total += element.doubleValue();
    return total;
} // fim do método sum
} // fim da classe WildcardTest
```

Paulo André Castro

POO

ITA - Stefanini 52

Resultado - Classe que utiliza Wildcard

```
integerList contains: [1, 2, 3, 4, 5]
Total of the elements in integerList: 15
```

```
doubleList contains: [1.1, 3.3, 5.5]
Total of the elements in doubleList: 9,9
```

```
numberList contains: [1, 2.4, 3, 4.1]
Total of the elements in numberList: 10,5
```

Paulo André Castro

POO

ITA - Stefanini 53

Criação de Arquivos JAR

- Java ARchive: Arquivos JAR são um meio simples de empacotar e distribuir aplicativos Java

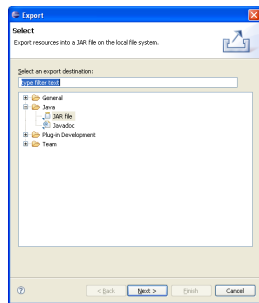
Paulo André Castro

POO

ITA - Stefanini 54

Arquivos JAR

- Criação
 - Opção Export no Popup Menu do Projeto
 - Selecionar Java | JAR File
 - Next



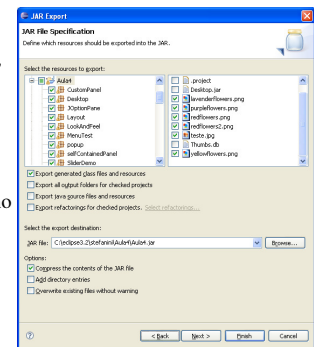
Paulo André Castro

POO

ITA - Stefanini 56

Criação de Arquivos JAR

- Selecione os recursos (imagens, arquivos Java, etc.) que devem estar no projeto
 - Atenção para selecionar também bibliotecas utilizadas
- Escolher nome e caminho do arquivo a ser gerado



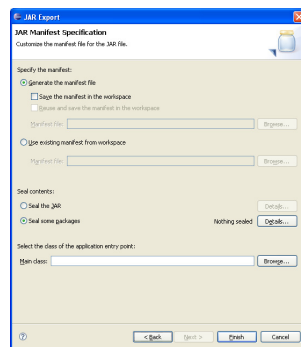
Paulo André Castro

POO

ITA - Stefanini 56

Criação de Arquivos JAR

- Pacotes Selados (Sealed Packages): Pacotes que devem obrigatoriamente estar no mesmo arquivo Jar, evita que se utilize erroneamente outras implementações que estiverem no classpath
- Arquivo Manifest gerado automaticamente ou incluído
- Definição de classe principal (classe com método main a ser executado inicialmente)



Paulo André Castro

POO

ITA - Stefanini 57

5.2. Networking

Sumário

- 5.2.1 Introdução
- 5.2.2 Manipulando URLs
- 5.2.3 Lendo um arquivo em um Web Server
- 5.2.4 Criando um Servidor Simples usando Stream Sockets
- 5.2.5 Criando um Cliente Simples usando Stream Sockets
- 5.2.6 Uma Aplicação Client/Server com Stream Socket
- 5.2.7 Segurança e Rede
- 5.2.8 Design Patterns usados nos Packages java.io e java.net
 - 5.2.8.1 Creational Design Patterns
 - 5.2.8.2 Structural Design Patterns
- 5.2.9 Javadoc: Documentação automática em java

Paulo André Castro

POO

ITA - Stefanini 58

5.2.1 Introduction

- Networking package is `java.net`
 - Socket-based communications
 - Applications view networking as streams of data
 - Connection-based protocol
 - Uses TCP (Transmission Control Protocol)
 - Packet-based communications
 - Individual packets transmitted
 - Connectionless service
 - Uses UDP (User Datagram Protocol)

Paulo André Castro

POO

ITA - Stefanini 59

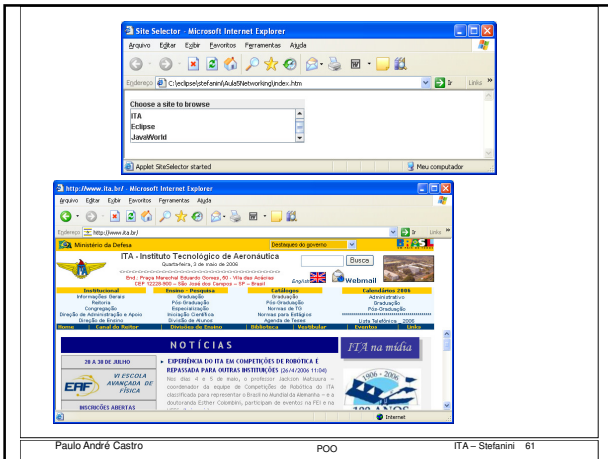
5.2.2 Manipulating URLs

- HyperText Transfer Protocol (HTTP)
 - Uses URIs (Uniform Resource Identifiers) to locate data
 - URIs frequently called URLs (Uniform Resource Locators)
 - Refer to files, directories and complex objects
- Exemplo do Uso da manipulação de URI em Applets

Paulo André Castro

POO

ITA - Stefanini 60



```

1 <html>
2 <title>Site Selector</title>
3 <body>
4 <applet code = "SiteSelector.class" width = "300" height = "75">
5   <param name = "title0" value = "Java Home Page">
6   <param name = "location0" value = "http://java.sun.com/">
7   <param name = "title1" value = "ITA">
8   <param name = "location1" value = "http://www.ita.br/">
9   <param name = "title2" value = "Eclipse">
10  <param name = "location2" value = "http://www.eclipse.org/">
11  <param name = "title3" value = "JavaWorld">
12  <param name = "location3" value = "http://www.javaworld.com/">
13 </applet>
14 </body>
15 </html>

```

Declare param tags for the applet

Paulo André Castro POO ITA - Stefanini 62

```

1 // SiteSelector.java
2 // This program uses a button to load a document from a URL.
3 import java.net.*;
4 import java.util.*;
5 import java.awt.*;
6 import java.applet.AppletContext;
7 import javax.swing.*;
8 import javax.swing.event.*;
9
10 public class SiteSelector extends JApplet {
11     private HashMap sites; // site names and URLs
12     private Vector siteNames; // site names
13     private JList siteChooser; // list of sites to choose from
14
15     // read HTML parameters and set up GUI
16     public void init()
17     {
18         // create HashMap and Vector
19         sites = new HashMap();
20         siteNames = new Vector();
21
22         // obtain parameters from HTML document
23         getSitesFromHTMLParameters();
24     }

```

Create HashMap and Vector objects

Paulo André Castro POO ITA - Stefanini 63

```

25 // create GUI components and layout interface
26 Container container = getContentPane();
27 container.add(new JLabel("Choose a site to browse"),
28 BorderLayout.NORTH);
29
30 siteChooser = new JList(siteNames);
31 siteChooser.addListSelectionListener(
32     new ListSelectionListener() {
33
34         // go to site user selected
35         public void valueChanged(ListSelectionEvent event)
36         {
37             // get selected site name
38             Object object = siteChooser.getSelectedValue();
39
40             // use site name to locate corresponding URL
41             URL newDocument = (URL) sites.get(object);
42
43             // get reference to applet container
44             AppletContext browser = getAppletContext();
45
46             // tell applet container to change pages
47             browser.showDocument(newDocument);
48         }
49     }
50

```

Method valueChanged goes to the selected Web site

Create the document

Show the document in the browser

Paulo André Castro POO ITA - Stefanini 64

```

51     } // end inner class
52     }; // end call to addListSelectionListener
53     container.add(new JScrollPane(siteChooser),
54 BorderLayout.CENTER);
55 // end method init
56
57 // obtain parameters from HTML document
58 private void getSitesFromHTMLParameters()
59 {
60     // look for applet parameters in HTML document and add to HashMap
61     String title, location;
62     URL url;
63     int counter = 0;
64
65     title = getParameter("title" + counter); // get first site title
66
67     // loop until no more parameters in HTML document
68     while (title != null) {
69
70         // obtain site location
71         location = getParameter("location" + counter);
72     }
73
74
75

```

Get Web site title

Get Web site location

Paulo André Castro POO ITA - Stefanini 65

```

76 // place title/URL in HashMap and title in Vector
77 try {
78     url = new URL(location); // convert location to URL
79     sites.put(title, url); // put title/URL in HashMap
80     siteNames.add(title); // put title in Vector
81 }
82 // process invalid URL format
83 catch (MalformedURLException urlException) {
84     urlException.printStackTrace();
85 }
86
87 ++counter;
88 title = getParameter("title" + counter); // get next site title
89
90 // end while
91 // end method getSitesFromHTMLParameters
92 // end class SiteSelector

```

Create URL of location

Add URL to HashMap

Add title to Vector

Get next title from HTML document

Paulo André Castro POO ITA - Stefanini 66

Exercícios

- Mudar as páginas e títulos para três páginas de sua escolha
- Substituir o componente JList por três botões que façam a operação de redirecionar a página

Projeto I - POO

- Crie um programa gráfico baseado em Swing com Barra de Menu, capaz de:
 - Barra de Menu deverá ter pelo menos as opções: Arquivo | Abrir & Sair e Ajuda | Sobre
 - No Menu Arquivo após a opção Sair, listar os últimos arquivos abertos (caso exista)
 - Na opção Abrir: abrir arquivos com extensão .zip.
 - Liste as entradas do arquivo .zip aberto na Interface Gráfica
 - Caso o arquivo selecionado seja um arquivo texto (.txt) apresente o conteúdo deste na janela, ao ser clicado duas vezes sobre o nome do arquivo.
 - Caso o arquivo selecionado seja um arquivo de imagem (.png, .jpg, .gif) apresente o conteúdo deste na interface gráfica
 - Caso não seja um arquivo texto nem imagem apresentar mensagem, informando ao usuário que não pode abrir arquivos binários.
 - Apresente no Caption da Janela: O nome do programa [– O nome do arquivo aberto caso haja algum.]
 - Na opção Ajuda | Sobre: Apresentar uma caixa de diálogo com informações (inclusive fotos) sobre o programa e seus autores.

Projeto I - POO

- Classes (e Interfaces) úteis para implementar o projeto I
 - JFrame
 - JList
 - JTextArea
 - ActionListener
 - Java.awt.FileDialog
 - Java.io.ZipInputStream, ZipOutputStream
- Diretrizes
 - Criar classe específica para Janela
 - Criar classe específica para realizar tratamento de eventos do programa
 - Criar classe específica para Guardar informações sobre o aplicativo: Nome do arquivo zip aberto, Nome do arquivo cujo texto é apresentado, Lista dos últimos quatro arquivos .zip abertos
 - Obedecer conceitos de OO: encapsulamento, polimorfismo, herança
 - Seguir boas práticas de programação: convenção de nomes, comentários (javadoc), etc.

Projeto I - POO

- Enviar ao professor através de email:
 - Projeto Eclipse Completo que implementa o Projeto I
 - Arquivos java e class
 - Projeto completo executável em um arquivo .jar
 - Relatório Simples (1 ou 2 páginas) formato(.doc, .rtf, ou txt), contendo:
 - Título
 - Nomes dos Alunos (Trabalho em dupla)
 - Resultados Obtidos
 - Comentários sobre o Projeto
 - Principais Dificuldades Encontradas
 - Conhecimentos Adquiridos, sugestões, etc.
- Prazo: Entrega via e-mail até dia 17/12