

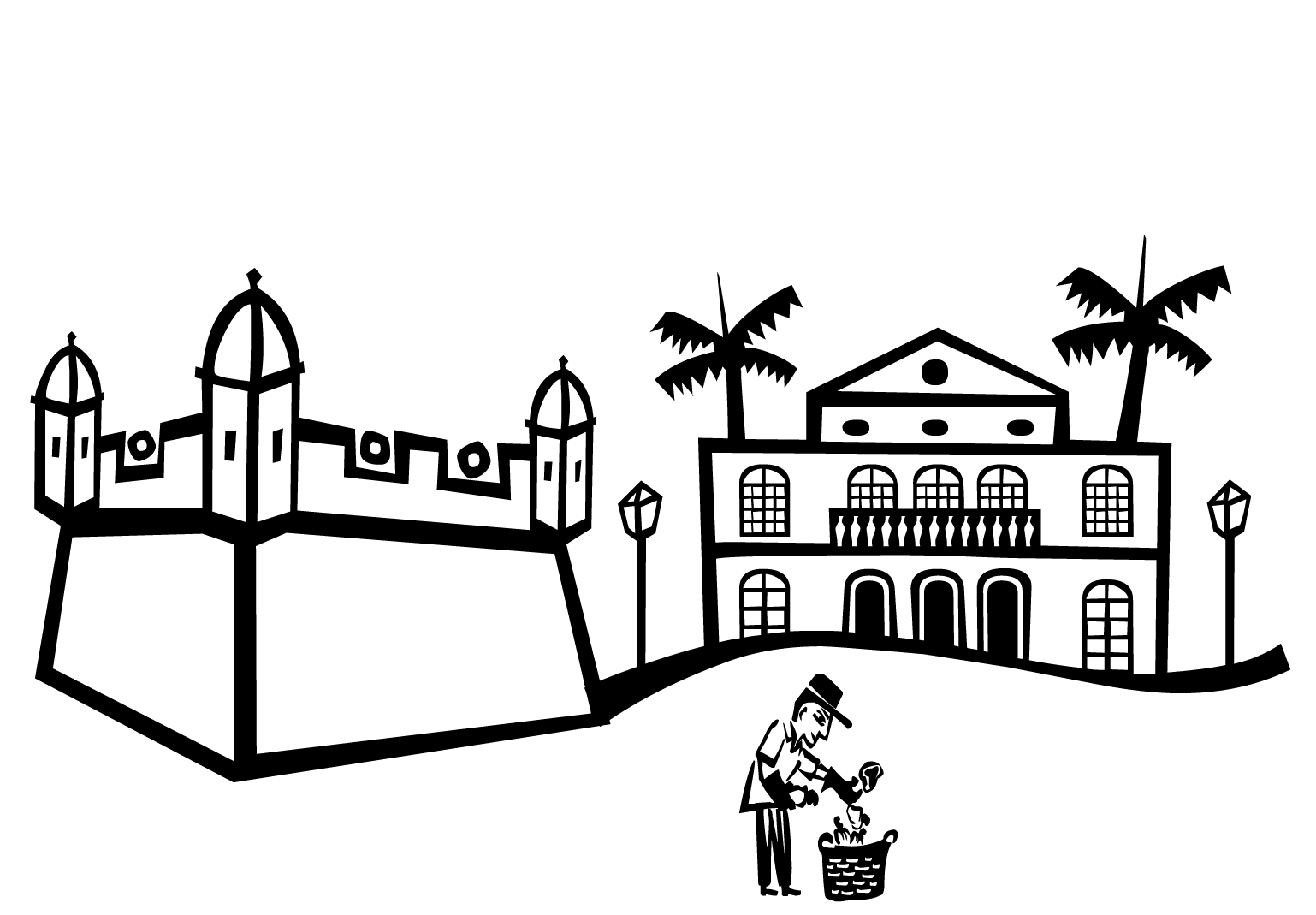
****

**Gabarito - Java Básico**

**Módulo 8**

1. Em relação ao tratamento de exceções na linguagem Java, das opções seguintes, aquela que **NÃO** está relacionada com este assunto é:
2. throws
3. finally
4. catch
5. try
6. **transient**
7. Analise as seguintes afirmações relacionadas aos conceitos básicos de programação e linguagens de programação orientadas a objetos.
8. Na programação orientada a objetos, o overflowem operações aritméticas e a divisão por zero não podem ser tratados como exceções.
9. Uma vez que uma exceção é disparada, o controle não pode retornar diretamente ao ponto de disparo.
10. Uma exceção termina o bloco no qual ela ocorreu.
11. O tratamento de exceções é utilizado para tratar erros de sintaxe, isto é, erros que acontecem como resultado da compilação de um programa.

As afirmativas verdadeiras são:

1. A e B
2. **B e C**
3. C e D
4. A e C
5. B e D
6. Indique qual será a saída do programa abaixo:

class TesteExcecao {

public static void main (String[] args) {

try {

int len = args.length;

System.out.println(args[len\*2]);

}

catch (ArithmeticException e) {

System.out.println("1");

}

catch (RuntimeException e) {

System.out.println("2");

}

catch (Exception e) {

System.out.println("3");

}

finally {

System.out.println("4");

}

System.out.println("Fim");

}

}

1. 3, 4, *Fim*
2. 2, 4
3. **2, 4, *Fim*** (a exceção causada, *ArrayIndexOutOfBounds,* por ter como classe base *RuntimeException*, é capturada na cláusula referente a esta exceção, imprimindo o valor 2. Em seguida, o fluxo segue para o bloco *finally* e imprime o valor 4. Por fim, é impresso o termo *Fim*.)
4. 3, 4
5. Escreva um programa utilizando os conceitos de Java aprendidos até aqui. Ele deverá conter um método que receba dois valores numéricos e some-os retornando o resultado. Os números de entrada deverão ser inseridos pelo usuário e lidos por meio da classe *Scanner*. Crie o tratamento de exceções para o caso de o usuário inserir um valor não numérico, imprimindo os detalhes do erro por meio do método *printStackTrace()*.

**import** java.util.Scanner;

**public** **class** Excecoes {

**public** **static** **void** main(String[] args) {

Scanner input = **new** Scanner(System.***in***);

**try** {

System.***out***.print("Digite o primeiro número: ");

**double** primeiro = input.nextInt();

System.***out***.print ("Digite o segundo número: ");

**double** segundo = input.nextInt();

**double** resultado = segundo + primeiro;

System.***out***.println("o resultado é: " + resultado);

} **catch** (InputMismatchException e) {

e.printStackTrace();

}

}

}

1. Desenvolva um programa que contenha uma classe chamada *NumeroMenorQueDezException*. Ela deverá receber como parâmetro em seu construtor um mensagem de tipo *String*. Na classe principal (onde está localizado o método *main*) será necessário criar um método chamado *verificarNumero*, que receberá como parâmetro um inteiro e que verificará se tal parâmetro é menor que 10. Caso positivo, este método deverá lançar uma exceção do tipo *NumeroMenorQueDezException*, contendo a mensagem “Número menor que 10”. Para testar seu código, crie no método principal uma variável que receba o valor 7 e passe-a como parâmetro para o método *verificarNumero*, não esquecendo de aplicar a este trecho de código o correto tratamento de exceções.

public class Numeros {

public static void main(String[] args) {

Numeros numeros = new Numeros();

int numero = 7;

try {

numeros.verificaNumero(numero);

} catch (NumeroMenorQueDezException e) {

System.out.println("Numero menor que 10.");

} finally {

System.out.println("Fim do código.”)

}

}

public void verificaNumero(int numero) throws NumeroMenorQueDezException{

if (numero < 10) throw new NumeroMenorQueDezException("Número menor que 10");

}

class NumeroMenorQueDezException extends Exception {

public NumeroMenorQueDezException(String string) {

super(string);

}

}

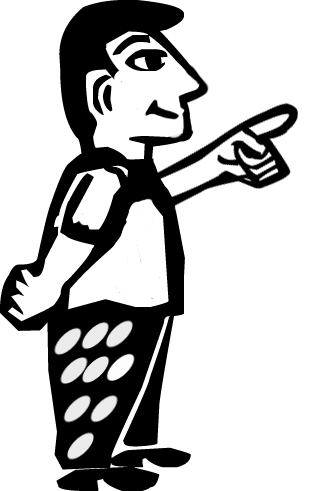
}

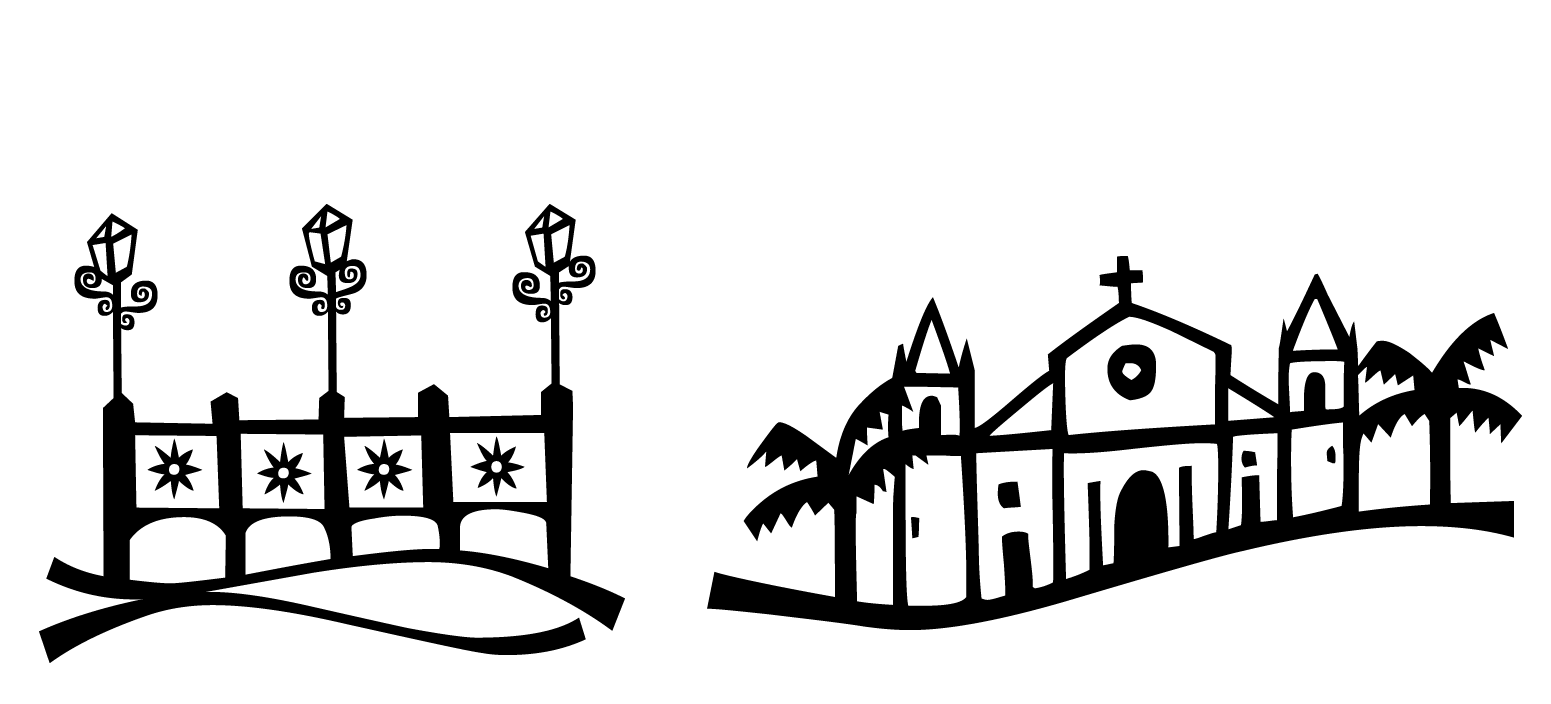








****

.

