Disciplina: Programação para Web I	Semestre: 3º
Turma: Noite	
Data: 21/03/2017	Professora: Silvia Bertagnolli

LISTA DE EXERCÍCIOS

1) Determine o que ocorre quando o código abaixo é executado:

```
public class Questao1 {
  public static void main(String[] args) {
         Telefone telefones[] = new Telefone[5];
         telefones[0] = new Telefone();
         telefones[2] = new Telefone(51, 55667788);
         for (int i = 0; i < telefones.length; i++) {
                System.out.println(telefones[i].toString());
         }
  }
}
```

2) Determine o que ocorre quando o código abaixo é executado:

```
import java.util.*;
public class Questao2 {
  public static void main(String[] args) {
         LinkedList<Telefone>  telefones = new LinkedList<Telefone>();
         telefones.add(new Telefone());
         telefones.add(new Telefone(51, 55667788));
         telefones.add(3, new Telefone(55, 33445566));
         telefones.remove(0);
         telefones.remove(0);
         telefones.remove(1);
  }
}
```

3) Determine o que ocorre quando o código abaixo é executado:

```
1. public class Questao3{
2.    public static void main(String args[]) {
3.      int i = 5571;
4.      i = i / 0;
5.      System.out.println("O resultado " + i);
6.    }
7.}
```

4) Determine o que ocorre quando o código abaixo é executado:

```
1. public class Questao4 {
2.    public static void main(String[] args) {
3.       try {
4.          int i = (int) Math.random();
5.          int j = 10 / i;
6.       }catch (ArithmeticException e) {
7.          e.printStackTrace();
8.       }
9.   }
10.}
```

5) Analisando a saída gerada, responda os itens abaixo para as questões 1 a 4:

4.1 Qual o nome da classe em que foi gerada a exceção?
4.2 Qual o nome do método em que a exceção foi gerada?
4.3 Qual a linha que gerou a exceção?
4.4 Qual exceção foi gerada?
4.5 Qual a causa da exceção gerada?

6) Faça o tratamento das exceções das questões 1 a 4.

7) Faça o tratamento das exceções do código abaixo.

```
1. import java.util.*;
2. public class Questao7{
3.   public static void main(String args[]) {
4.     TreeSet<Number> numeros = new TreeSet<>();
5.     numeros.add(new Integer(5));
6.     numeros.add(8);
7.     numeros.add(new Double(5.9));
8.     numeros.add(null);
9.     for(Number num : numeros)
10.      System.out.println("Num: " + num);
11.    }
12. }
```

8) Analise o código abaixo, determine o que ele irá imprimir e após faça o tratamento das exceções.

```
1. public class Questao8 {
2.  public static void main(String[] args) {
3.    System.out.println("Início do main");
4.    metodo1();
5.    System.out.println("Fim do main");
6.  }
7.  static void metodo1() {
8.    System.out.println("Início do metodo1");
9.    metodo2();
10.   System.out.println("Fim do metodo1");
11. }
12. static void metodo2() {
13.    System.out.println("Início do metodo2");
14.    int[] array = new int[10];
15.    for (int i = 0; i <= 15; i++) {
16.       array[i] = i;
17.     System.out.println(i);
18.   }
19.   System.out.println("Fim do metodo2");
20.  }
21. }
```

9) Analise o código abaixo, após determine o que ele irá imprimir. Agora, responda aos itens 9.1 a 9.5:

9.1 O que o código da linha 10 faz?
9.2 O que o código da linha 14 faz?
9.3 Você pode alterar a ordem dos blocos catch da linha 6 e da linha 8? Justifique sua resposta.
9.4 Você pode alterar a ordem dos blocos catch da linha 8 e da linha 11? Justifique sua resposta.
9.5 Você pode alterar a ordem dos blocos catch da linha 11 e da linha 13? Justifique sua resposta.

```
1. public class Questao9 {
2.    public static void main(String args[]) {
3.      try{
4.        int a[] = new int[10];
5.        System.out.println("Acessando elemento 20 : " + a[20]);
6.      }catch(NullPointerException e){
7.         System.out.println("NullPointerException :" + e.getMessage());
8.      }catch(ArrayIndexOutOfBoundsException e){
9.         System.out.println("ArrayIndexOutOfBoundsException :" +
10.        e.getMessage());
11.     }catch(RuntimeException e){
12.       System.out.println("RuntimeException :" + e.getMessage());
13.     }catch(Exception e){
14.       e.printStackTrace();
15.     }catch(Error e){
16.       System.out.println("Error :" + e.getMessage());
17.     }catch(Throwable e){
18.       System.out.println("Throwable :" + e.getMessage());
19.     }
20.   }
21. }
```

10) Analise o código abaixo, após determine o que ele irá imprimir. Justifique sua resposta.

```
1. public class Questao10 {
2.    public static Object criaObjeto() { return null; }
3.    public static void main(String[] args) {
4.      try {
5.        Object obj = criaObjeto();
6.        System.out.println(obj.toString());
7.      }catch (NullPointerException e) {
8.         e.printStackTrace();
9.      }
10.   }
11. }
```

11) Explique com suas palavras a diferença entre os blocos, try, catch e finally. Você acha que um bloco catch pode ser definido sem um bloco try? Você acha que um bloco finally pode ser definido sem um bloco try? Você acha que um bloco try pode ser definido sem um bloco catch?

12) Analisar o programa a seguir para determinar a saída gerada quando a entrada realizada pelo usuário for 2 e, posteriormente, 3.

```
1.  import java.io.*;
2.  import javax.swing.*;
3.  public class Questao12 {
4.    public static void main(String[] args) {
5.     System.out.println("1");
6.     try{
7.        System.out.println("2");
8.        int num = Integer.parseInt(JOptionPane.showInputDialog(null,
9.                                                "Informe um número:"));
10.       System.out.println("Número:"+num);
11.       switch(num){
12.           case 1: System.out.println("3");
13.                   int valor1 = Integer.parseInt("abc");
14.                   System.out.println(valor1);
15.           case 2: System.out.println("4");
16.                   String palavras[] = {"casa", "apto", "morada"};
17.                   System.out.println(palavras[3]);
18.           case 3: System.out.println("5");
19.                   double vetor[] = null;
20.                   for (double valor: vetor) {
21.                        System.out.println(valor);
22.                   }
23.           case 4: System.out.println("6");
24.                   File f = new File("c:\\teste.txt");
25.                   f.createNewFile();
26.       }
27.    }catch(NumberFormatException e){
28.           System.out.println("3.1");
29.    }catch(ArrayIndexOutOfBoundsException e){
30.           System.out.println("4.1");
31.    }catch(NullPointerException e){
32.           System.out.println("5.1");
33.    }catch(IOException e){
34.           System.out.println("6.1");
35.    }finally{
36.           System.out.println("7");
37.    }
38.   }
39. }
```

13) Determine a saída gerada pelo código abaixo.

```
1. class ProprioErro extends Exception {
2.    private int erro;
3.    ProprioErro (int erro) { this.erro =erro; }
4.    public String toString () { return("ProprioErro[" +erro+ "]");}
5. }

1. public class Teste{
2.    static void calcula (int a) throws ProprioErro {
3.         if(a<=12) throw new ProprioErro(a);
4.    }
5.    public static void main(String args[]){
6.      try {
7.             calcula(9);
8.             calcula(11);
9.      }catch (ProprioErro e){
10.         System.out.println("Erro encontrado: " +e);
11.     }
12.  }
13.}
```

14) Crie a interface genérica Lista como descrito abaixo:

- A lista irá conter elementos

- Defina os métodos abaixo usando genéricos onde for possível:

public void adicionar(int indice, Object obj)
public boolean remover(Object obj)
public String listar()
public int totalizar()
public Object getFirst()
public Object getLast()

15) Defina a classe MinhaLista observando as seguintes regras:

- MinhaLista deve ser definida como genérica e usar a interface Lista (Questão 14)

- Use a classe LinkedList para armazenar objetos na classe MinhaLista

- Use todos os conceitos de genéricos e observe os conceitos de tratamento de exceções

- Defina os métodos adicionar, remover e getUltimo considerando as observações abaixo:

1. `public void adicionar(int indice, Object obj)- método deve tratar a exceção definida pela documentação`
2. `public Object remover(Object obj) – método deve propagar a exceção definida pela documentação`
3. `public int getUltimo() – método deve causar a exceção definida pela documentação`
4. `public String toString()`

16) Crie uma classe de Testes para testar a lista definida na questão 15. Essa classe deve fazer os tratamentos das exceções, quando for necessário.