| **Disciplina:** Programação para Web I | Semestre: 3º |
|---|---|
| Professora: Silvia Bertagnolli | |

# LISTA DE EXERCÍCIOS

**1) Faça o tratamento das exceções abaixo. Agora, execute o programa 2 vezes, o que acontece? Explique com suas palavras o que ocorreu. Com base no que ocorreu faça novamente o tratamento das exceções.**

```
1.          FileSystem fs = FileSystems.getDefault();
2.          Path diretorio = fs.getPath("C:", "Diretorio", "Teste");
3.          Path arquivo = diretorio.resolve("Teste.txt");
4.          Files.createDirectories(diretorio);
5.          Files.createFile(arquivo);
```

**Observações:**

1. O objeto **Path** usado em conjunto com a classe **Files** pode criar ou deletar um arquivo ou diretório
2. O objeto **Path** representa o diretório que será criado passando três parâmetros para o método **getPath()** de **FileSystem**. O primeiro é a raiz do sistema de arquivos, o segundo e o terceiro representam a estrutura de diretórios que será construída.

**2) Análise o código abaixo e faça o tratamento das exceções. Agora, faça o código para ler os valores que foram salvos no arquivo:**

```
public class Questao2 {
    public static void main(String[] args) {
       FileSystem fs = FileSystems.getDefault();
       List<String> list = criarListaString();
       Path diretorio = fs.getPath("C:", "Diretorio", "Teste");
       Path arquivo = diretorio.resolve("teste.txt");
       Charset charset = Charset.forName("UTF-8");
       Files.write(arquivo, list, charset, StandardOpenOption.APPEND);
       lerArquivo(arquivo, charset);
    }
    public static List<String> criarListaString () {
        List<String> list = new ArrayList<String>();
        list.add("ABC"); list.add("DEF");
        list.add("GHI"); list.add("JKL");
        list.add("MNO");
        return list;
    }
}
```

1. Obs.: para fazer a leitura procure na classe Files um método que lê todas as linhas

**3) Faça o tratamento das exceções abaixo. Agora, execute o programa e análise o seu comportamento**

```
public class Questao3 {
   public static void main(String[] args) {
        Path txt = Paths.get("teste.txt");
        BasicFileAttributes info = Files.readAttributes(txt, BasicFileAttributes.class);

        System.out.format("Data de criação: %s%n", info.creationTime());
        System.out.format("Último acesso: %s%n", info.lastAccessTime());
        System.out.format("Última modificação: %s%n", info.lastModifiedTime());
        System.out.format("É um diretorio: %s%n", info.isDirectory());
        System.out.format("Tamanho: %s bytes%n", info.size());
   }
}
```

**Observações:** A interface BasicFileAttributes permite acessar atributos básicos de um arquivo dentro do sistema

**4) Identificar quais métodos geram exceção no código abaixo. Após, determinar se elas são ou não verificadas pelo compilador. Agora, faça o tratamento das exceções do código abaixo:**

```
public class Questao4 {
      public static void main(String[] args) {
            Scanner entrada = new Scanner(System.in);
            System.out.printf("Informe o número para a tabuada:\n");
            int n = entrada.nextInt();
            System.out.printf("Informe a pasta:\n");
            String pasta = entrada.nextLine();
            FileWriter arq = new FileWriter(pasta+"\\tabuada.txt");
            PrintWriter gravarArq = new PrintWriter(arq);
            gravarArq.printf("+--Resultado--+%n");
            for (int i = 1; i <= 10; i++) {
                  gravarArq.printf("| %2d X %d = %2d |%n", i, n, (i * n));
            }
            gravarArq.printf("+-------------+%n");
            arq.close();
            System.out.println("\nTabuada do " + n + " foi gravada na " +
                                                pasta + "\tabuada.txt\n");
      }
}
```

**5) Analisar o programa abaixo e verifique o que ele faz. Agora, faça o tratamento das exceções que podem ser geradas pelo código.**

```
public class Questao5{
      public static void main(String[] args) {
            //gravando caracteres e Strings
            File arquivoEscrita = new File("Ex5.txt");
            FileWriter fw = new FileWriter(arquivoEscrita) ;
            fw.write('2');
            fw.write("2");
            fw.flush();
            fw.close();
            //gravando caracteres e Strings
            File arquivoLeitura = new File("Ex5.txt");
            FileReader fr = new FileReader(arquivoLeitura);
            int  c =   fr.read();
            while( c != -1){
                 System.out.print( (char) c );
                 c =  fr.read();
            }
      }
}
```

**Observações:**

1. as classes FileWriter e FileReader são usadas para gravar e ler um fluxo de caracteres de um arquivo
2. a classe FileReader nos fornece o método read que lê um único caractere do arquivo e retorna o número inteiro de seu código na tabela unicode, ou se for o final do arquivo ele retornará -1

**6) Explique com suas palavras a diferença entre o código da Questão 5 e o código abaixo, no que diz respeito à leitura dos dados. Após, faça o tratamento das exceções do código abaixo.**

```
            File arquivo = new File("teste.txt");
            FileReader fr = new FileReader(arquivo);
            char[]  c =  new char[4];
            fr.read(c);
            System.out.print( c );
            fr.close();
```

**7) Analisar o programa abaixo e verifique o que ele faz. Agora, faça o tratamento das exceções que podem ser geradas pelo código.**

```
public class Questao7{
      public static void main(String[] args) {
            File arquivo = new File("Ex7.txt");
            FileWriter fw = new FileWriter( arquivo ) ;
            BufferedWriter escrita = new BufferedWriter(fw);
            escrita.write( "teste1" );
            escrita.newLine();
            escrita.write( "teste2");
            escrita.flush();
            escrita.close();

            FileReader fr = new FileReader(arquivo);
            BufferedReader leitura = new BufferedReader(fr);
            String content;
            while( ( content = leitura.readLine() ) != null){
                System.out.println( content );
            }
            leitura.close();
      }
}
```

**Observações:**

1. As classes BufferedWriter e BufferedReader são usadas, respectivamente, para escrever e ler caracteres usando um buffer

**8) Analisar o programa abaixo e verifique o que ele faz. Agora, faça o tratamento das exceções que podem ser geradas pelo código.**

```
public class Questao8{
  public static void main(String[] args) {
    File arquivo = new File("Ex8.txt");
    PrintWriter escrita = new PrintWriter(arquivo);
    escrita.println(true);
    escrita.println(10);
    escrita.println(10.20);
    escrita.println("teste");
    escrita.printf("str: %s | double: %.2f | int: %5d " , "teste", 10f , 200);
    escrita.close();
  }
}
```

**Observações:**

1. A classe PrintWriter possui os métodos println e printf que gravam em um arquivo os dados passados como parâmetro

**9) Analisar o programa abaixo e verifique o que ele faz. Agora, faça o tratamento das exceções que podem ser geradas pelo código.**

```
public class Questao9{
      public static void main(String[] args) {
            File arquivo = new File("Ex9.bin");
            //gravando fluxo baseados em bytes
            OutputStream saida = new FileOutputStream(arquivo);
            byte[] b = {50,51,52,53};
            String string = "Teste com várias palavras";
            saida.write( 53 );
            saida.write( b );
            saida.write( string.getBytes() );
            saida.flush();
            saida.close();

            //lendo fluxo baseados em bytes
            InputStream entrada = new FileInputStream(arquivo);
            int content;
            while ( (content = entrada.read() ) != -1) {
                System.out.println( content +" - "+ ( (char) content) );
            }
            entrada.close();
      }
}
```

**Observações:**

1. a classe FileOutputStream é usada para gravar bytes em um arquivo
2. a classe FileInputStream é usada para ler bytes de um arquivo
3. para escrever dados em um arquivo é usado o método write que pode receber um byte ou um vetor de bytes
4. o método getBytes converte os caracteres da String em bytes, pois a classe OutputStream precisa desse formato para que os bytes sejam gravados
5. o laço while é usado para percorrer o arquivo até o fim, onde o método read retorna -1 se chegar no final do arquivo.

**10) Analisar o programa abaixo e verifique o que ele faz. Agora, faça o tratamento das exceções que podem ser geradas pelo código.**

```
public class Questao10{
      public static void main(String[] args) {
        File arquivo = new File("Ex10.txt");
        //escrevendo dados
        FileOutputStream fo = new FileOutputStream( arquivo ) ;
        BufferedOutputStream escrita = new BufferedOutputStream(fo);
        escrita.write( "teste1".getBytes());
        escrita.write("\n".getBytes() );//inserindo um caractere de nova linha
        escrita.write( "teste2".getBytes());
        escrita.flush();
        escrita.close();
        //lendo dados
        FileInputStream fi = new FileInputStream(arquivo);
        BufferedInputStream entrada = new BufferedInputStream(fi);
        int content;
        while( ( content = entrada.read() ) != -1){
              System.out.println( content  + " - " + ( (char) content) );
        }
        entrada.close();
      }
}
```

**Observações:**

1. Na Questao9 o código trabalha gravando e lendo de arquivos, byte por byte ou caractere por caractere
2. Na Questao10 trabalha gravando uma grande quantidade de dados de uma única vez
3. As classes BufferedOutputStream e BufferedInputStream são usadas, respectivamente, para escrever e ler dados em um arquivo usando um buffer

**11) Faça o tratamento de exceções da classe abaixo e, após, crie uma classe de testes que use a classe Arquivo e que salve objetos do tipo Telefone. Faça as correções que julgar necessárias para que o código utilize genéricos e funcione corretamente.**

```
1.  import java.io.*;
2.  public class Arquivo{
3.    private ObjectOutputStream saida;
4.    private ObjectInputStream entrada;
5.    private String nomeArq;
6.
7.    public Arquivo(String nome){ nomeArq=nome; }
8.
9.    public void abrir(String tipo){
10.     if (tipo.equals("w")){ // abre para gravação
11.           saida = new ObjectOutputStream(new FileOutputStream( nomeArq) );
12.     System.out.println("Aberto para Gravação");
13.     }
14.     else{ // abre para leitura
15.         entrada = new ObjectInputStream(new FileInputStream( nomeArq) );
16.         System.out.println("Aberto para Leitura");
17.     }
18.   }
19.   public void gravarObjeto(Object obj){
20.       saida.writeObject(obj);
21.       saida.flush();
22.   }
23.   public Object lerObjeto(){
24.       return entrada.readObject();
25.   }
26.   public void fechar(){
27.       saida.close();
28.       entrada.close();
29.   }
30. }
```