Classe File - Propriedades e Métodos

Programação Orientada a Objetos

Introdução

- Iniciamos com uma discussão sobre a entrada e saída de arquivos usando a classe java.io.File;
- Exemplos de métodos construtores;
- Aŕesentação e uso dos principais métodos;
- Exemplos e Exercícios.

O Pacote java.io

- **Import java.io.***
- Disponibiliza recursos para a entrada e saída de dados através de fluxos de dados, serialização e o sistema de arquivos (arquivos, diretórios e seus dados).
- Também oferece recursos para manipulação de dados durante o processo de leitura e gravação.

A classe java.io.File

- Usada para representar o sistema de arquivos.
- A existência de um objeto não significa a existência de um arquivo ou diretório.
- Contém métodos para testar a existência de arquivos, apagar arquivos, criar diretórios, listar o conteúdo de diretórios, etc..

A classe File

- Construtores Públicos:
  - public File(String path);
    - Cria uma nova instância a partir de um caminho do completo ‘path’.
  - public File(String path, String name);
    - Cria uma nova instância a partir da união do caminho ‘path’ com o ‘name’.
  - public File(File dir, String name);
    - Cria uma nova instância a partir da união do caminho do objeto ‘dir’ com o ‘name’.

Uso de contrabarra ‘\’ em nomes de arquivos

- Ao especificar um caminho ou nome de arquivo como uma String, e o nome contém caracteres de contrabarra ‘\’, deve ser usada duas contrabarras:
  - Exemplo: File inputFile = new File("c:\\homework\\input.dat");

Uso de contrabarra ‘\’ em nomes de arquivos

- Caso use ‘\’ é considerado como o caractere e pode ser associado com o caractere seguinte para formar um significado especial, como ‘\n’ que significa quebra de linha.
- A combinação ‘\\’ é que representa uma única contrabarra.
- Mas quando um usuário fornece um nome de arquivo a um programa, entretanto, o usuário não deve digitar a barra invertida duas vezes.
- Alternativa usar a barra ‘/’ como caractere separador.

A classe File - Alguns dos métodos:

- *String* getName()
- *String* getAbsolutePath()
- *String* getParent()
- *long* lastModified()
- *long* length()
- *String[]* list()
- *File[]* listFiles()

- *boolean* exists()
- *boolean* canWrite()
- *boolean* canRead()
- *boolean* isFile()
- *boolean* isDirectory()
- *boolean* delete()
- *boolean* mkdir()

https://docs.oracle.com/javase/7/docs/api/java/io/File.html

A classe File - Exemplo 01:

```
import java.io.File;
import java.io.IOException;
/**
 * @author fabricio@utfpr.edu.br
 */
public class TestaArquivo01 {
    public static void main(String[] args) throws IOException {
        File diretorio = new File("/home/fabricio/temp/");
        if (diretorio.mkdir()) {
            System.out.println("Pasta criada com sucesso.");
        } else if (diretorio.exists()) {
            System.out.println("Pasta já existe.");
        } else {
            System.out.println("Algum problema. Pasta não foi criada.");
        }
    }
}
```

A classe File - Exemplo 02:

```
import java.io.File;
import java.util.Date;
/**
 * @author fabricio@utfpr.edu.br
 */
public class TestaArquivo02 {
    public static void main(String[] args) {
        File dir = new File("/home/fabricio/temp/");
        File subdir = new File(dir, "subdir1");
        subdir.mkdir();//implementar verificação.
        String[] listagem = dir.list();
        for (String arquivo : listagem) {
            File filho = new File(dir, arquivo);
            System.out.println("Nome: "+ filho.getName());
            System.out.println(filho.getAbsolutePath());
            System.out.println("É diretorio? " + filho.isDirectory());
            System.out.println("É arquivo? " + filho.isFile());
            System.out.println("Tamanho = " + filho.length() + " bytes.");
            Date aux = new Date(filho.lastModified());
            System.out.println("Última alteração = " + aux);
        }
        subdir.delete();//implementar verificação.
    }
}
```

A classe File - Exemplo 03:

```
import java.io.File;
import java.util.Date;
import java.util.Scanner;
/**
 * @author fabricio@utfpr.edu.br
 */
public class TestaArquivo03 {
    public static void leituraPasta(File dir) {
        String[] listagem = dir.list();
        for (String arquivo : listagem) {
            File filho = new File(dir, arquivo);
            System.out.println("Nome: "+ filho.getName());
            System.out.println(filho.getAbsolutePath());
            System.out.println("É diretorio? " + filho.isDirectory());
            System.out.println("É arquivo? " + filho.isFile());
            System.out.println("Tamanho = " + filho.length() + " bytes.");
            Date aux = new Date(filho.lastModified());
            System.out.println("Ultima alteração = " + aux+"\n\n");
        }
    }
    public static void main(String[] args) {
        System.out.println(" Entre com o caminho: ");
        Scanner leitura = new Scanner(System.in);
        String path = leitura.nextLine();
        File arquivo = new File(path);
        leituraPasta(arquivo);
    }
}
```

Exercício Acompanhado

1. Usar os exemplos explicados como base para desenvolver uma aplicação que dado uma pasta de entrada, calcular o espaço ocupado por essa pasta, identificando:
    a. Número total de pastas e subpastas;
    b. Número total de arquivos;
    c. Tamanho total ocupado.

Referências

- DEITEL, P.J. Java - Como Programar. Porto Alegre: Bookman, 2001.
- NIEMEYER, Patrick. Aprendendo java 2 SDK. Rio de Janeiro: Campus, 2000.
- MORGAN, Michael. Java 2 para Programadores Profissionais. Rio de Janeiro: Ciência Moderna, 2000.
- HORSTMANN, Cay, S. e CORNELL, Gary. Core Java 2. São Paulo: Makron Books, 2001 v.1. e v.2.