# Abrindo um arquivo para leitura e edição



#### Editar dados de um arquivo

- A classe DataOutputStream somente funciona com duas modalidades:
  - Recriar um arquivo
  - Acrescentar dados ao final do arquivo.
- Não é possível alterar dados já gravados com DataOutputStream
  - Não possui skip()



#### Classe RandomAccessFile

#### RandomAccessFile

- + RandomAccessFile(name : String, mode : String)
- + RandomAccessFile(file : File, mode : String)
- + read(): int
- + read(b : byte[]) : int
- + readFully(b : byte[]) : void
- + skip(n:int):int
- + write(b:int): void
- + write(b : byte[]) : void
- + length(): long
- + setLength(newLength : long) : void
- + close(): void
- + readBoolean(): boolean
- + readByte(): byte
- + readShort(): short
- + readChar(): char
- + readInt(): int
- + readLong() : long
- + readFloat(): float
- + readDouble(): double
- + readUTF(): String
- + writeBoolean(v : boolean) : void
- + writeByte(v:int): void
- + writeShort(v:int): void
- + writeChar(v:int): void
- + writeInt(v:int): void
- + writeLong(v : long) : void
- + writeFloat(v: float): void
- + writeDouble(v: double): void
- + writeUTF(str : String) : void

#### <<interface>> DataInput

- + readFully(b : byte[]) : void
- + skipBytes(n : int) : int
- + readBoolean(): boolean
- + readByte(): byte
- + readShort() : short
- + readChar() : char
- + readInt(): int
- + readLong() : long
- + readFloat() : float
- + readDouble() : double
- + readUTF(): String

#### <<interface>> DataOutput

- + write(b:int): void
- + write(b : byte[]) : void
- + writeBoolean(v : boolean) : void
- + writeByte(v:int): void
- + writeShort(v:int): void
- + writeChar(v:int): void
- + writeInt(v:int): void
- + writeLong(v : long) : void
- + writeFloat(v : float) : void
- + writeDouble(v : double) : void
- + writeBytes(s : String) : void
- + writeUTF(s : String) : void

#### RandomAccessFile

- O parâmetro mode pode ser:
  - **r** para acesso somente leitura.
  - rw para leitura e gravação.
  - rws para leitura e gravação. Cada gravação é aplicada imediatamente no arquivo. Causa a alteração da data de modificação do arquivo
  - **rwd** Similar ao **rws**, porém somente causa a alteração da data de modificação do arquivo quando o arquivo é fechado.
- Os modos rws e rwd são mais lentos.



## Principais métodos de RandomAccessFile

Membro	Descrição
skip (long)	Salta para uma posição a frente do <i>inputstream</i> , em relação à posição atual. Deve ser um valor positivo.
seek (long)	Salta para uma posição no inputStream. A posição é em relação ao primeiro byte do arquivo. Exemplo: independente da posição que o ponteiro estiver, seek(50) posiciona no 51° byte do arquivo.
setLength (long)	Permite truncar o arquivo. Sempre informar um valor menor do que o atual.



### Fechamento de streams



#### Fechamento de arquivo

Depois de manipulado o arquivo, ele deve ser fechado:

```
File arquivo = new File("D:\\Temp\\dados.bin");
FileInputStream fis = new FileInputStream(arquivo);

int dado;
while ((dado = fis.read()) != -1) {
    System.out.println( dado );
}

fis.close();
```

Se ocorrer erro na leitura de arquivo, esta linha não é executada



#### Fechamento de arquivo

```
File arquivo = new File("D:\\Temp\\dados.bin");
FileInputStream fis = null;

try {
    fis = new FileInputStream(arquivo);
    int dado;
    while ((dado = fis.read()) != -1) {
        System.out.println( dado );
    }
} finally {
    if (fis != null)
        fis.close();
}
```

```
try (FileInputStream fis = new FileInputStream(arquivo)) {
   int dado;
   while ((dado = fis.read()) != -1) {
       System.out.println( dado );
   }
}
```

Sintaxe resumida utilizando o *try with resources* 

Para usar try(...) os objetos instanciados devem ser de classes que implementam a interface java.lang.AutoCloseable

