Escreva uma aplicação para ler um arquivo binário origem e gravá-lo em outro arquivo (arquivo destino).

Requisitos:

- Os nomes dos arquivos (origem e destino) devem ser definidos na chamada da aplicação via argumento de linha de comando.
- A leitura e gravação devem ser realizadas em blocos de bytes e não byte a byte.
- Ao final, deve-se exibir o tempo total da cópia em milisegundos, caso a cópia tenha sido bem sucedida.
- Teste vários tamanhos de bloco e encontre um tamanho que ofereça um bom desempenho. Crie uma tabela para comparar os resultados encontrados.

Sobre a aplicação:

- Decidi deixar a aplicação mais genérica onde você passa o nome do arquivo de origem, destino e o tamanho de blocos de bytes a serem lidos.
- Deixei um arquivo "city.jpg" na pasta static do projeto para testarmos a taxa de transferência e o tempo em ms durante a cópia dos dados.
- Para rodar a aplicação basta dar "run" no arquivo App.java, ele abrirá o console e então digite os seguintes dados na mesma linha:

```
  arquivo-origem arquivo-destino tamanhoDosBlocosDeBytes

  Por exemplo:
    city.jpg city-copy.jpg 8152
```

- Quando executado terá como output os seguintes dados:

```
Transfer data to destiny file finish with success!
total time: ${time} ms
total size transferred: ${size} bytes
throughput: ${size/time} bytes/ms
```

Resultados

- Foi constatado que quando aumentamos o valor do tamanho dos blocos de bytes o tempo de

transferência tende a diminuir, ou seja o tamanho dos blocos é inversamente proporcional ao tempo gasto de leitura.

- Foi criado uma tabela onde é feito uma cópia do arquivo "city.jpg" com diferentes tamanhos de bloco de bytes.

Size of block bytes (bytes)	Time (ms)	Size of file (bytes)	Throughput (bytes/ms)
1	215859	17878139	82
100	2596	17878139	6886
1000	504	17878139	35472
8152	92	17878139	194327
10000	146	17878139	122453
100000	59	17878139	303019
1000000	58	17878139	308243
10000000	65	17878139	308243
17878139	66	17878139	270880

Conclusão

- Como visto na tabela, a medida que aumentamos a quantidade de bytes a serem transferidos há uma diminuição considerável do tempo, já que a cada transferẽncia será "puxado" uma maior quantidade de bytes.
- Acredito que dependendo da aplicação a ser desenvolvida e do host que irá executá-la poderá haver diferentes valores da quantidade de bytes a terem uma maior eficiência e compatibilidade.
- No meu computador, acredito que o tamanho dos blocos de bytes sendo 8152 conseguiu dar um excelente resultado, sem comprometer toda a memória da máquina.
- Ainda pela tabela perceba que a partir da linha 7 para as outras mais abaixo há um pequeno aumento no tempo de transferência isso se deve ao fato de quando a máquina tenta ler uma maior quantidade de bytes por vez os esforços para armazenar no array de bytes aumenta. Ou seja, nem sempre a melhor decisão seria um maior tamanho de bloco de bytes por transferência.