Qual classe seria a melhor para ler um arquivo binário em um objeto Java.

A. **ObjectReader**
B. **ObjectWriter**
C. **BufferedStream**
D. **ObjectOutputStream**
E. **FileReader**
F. **ObjectInputStream**
G. **Nenhuma das opções acima**

Análise da Questão

A. **ObjectReader**:
   - **Erro**: Não existe uma classe `ObjectReader` na API padrão do Java. Essa opção é inválida. Essa classe existe na API `jackson-databind` .

B. **ObjectWriter**:
   - **Erro**: Assim como `ObjectReader`, `ObjectWriter` também não existe na API padrão do Java. Essa classe existe na API `jackson-databind` .

C. **BufferedStream**:
   - **Erro**: `BufferedStream` também não é uma classe válida na API do Java. Existem `BufferedInputStream` e `BufferedOutputStream`, mas não uma classe `BufferedStream` específica. Na realidade `BufferedStream` é uma classe .NET

D. **ObjectOutputStream**:
   - **Erro**: `ObjectOutputStream` é usada para escrever objetos em um fluxo de saída (output), ou seja, para gravar dados. Não é utilizada para ler dados de um arquivo.

E. **FileReader**:
   - **Erro**: `FileReader` é usada para ler dados de arquivos de texto (caracteres). Como a pergunta é sobre ler arquivos binários, `FileReader` não é apropriada.

F. **ObjectInputStream**:
   - **Resposta Certa**: `ObjectInputStream` é usada para ler objetos de um fluxo de entrada (input). É a classe apropriada para ler um arquivo binário e deserializar (converter de volta) os dados para um objeto Java.

G. **Nenhuma das opções acima**:
   - **Erro**: A resposta correta é a F.

Resposta Correta:

F. **ObjectInputStream**

Vamos entender um pouco melhor sobre a classe `ObjectInputStream`.

O que é a classe `ObjectInputStream`?

A classe `ObjectInputStream` é usada em Java para ler objetos que foram previamente escritos em um fluxo (stream) de dados, geralmente usando a classe `ObjectOutputStream`. É como se fosse uma máquina que pega dados binários e os reconstrói em objetos Java, que você pode usar no seu programa.

Quando usar `ObjectInputStream`?

Você usa `ObjectInputStream` quando precisa ler objetos de um arquivo ou de uma rede (por exemplo, de um socket) que foram escritos usando `ObjectOutputStream`. Um exemplo comum é quando você salva o estado de um objeto em um arquivo e depois quer carregar esse estado de volta para continuar trabalhando com ele.

Como funciona?

1. **Escrita do Objeto**: Primeiro, você tem um objeto que deseja salvar. Você usa `ObjectOutputStream` para escrever esse objeto em um arquivo.

2. **Leitura do Objeto**: Depois, você usa `ObjectInputStream` para ler o objeto de volta a partir do arquivo.

Passo a Passo

Aqui está um exemplo simples para ilustrar como usar `ObjectInputStream`:

Passo 1: Escrevendo um objeto em um arquivo

```java
package outrostestes;

import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

// Crie uma classe que implementa Serializable
class Pessoa implements Serializable {
    private static final long serialVersionUID = 1L;
    String nome;
    int idade;

    Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }
}

public class EscreveObjeto {
    public static void main(String[] args) {
        Pessoa pessoa = new Pessoa("João", 30);

        try (FileOutputStream fileOut = new FileOutputStream("pessoa.ser");
             ObjectOutputStream out = new ObjectOutputStream(fileOut)) {

            out.writeObject(pessoa); // Escreve o objeto no arquivo
            System.out.println("Objeto salvo em pessoa.ser");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Escrevendo o Objeto (`EscreveObjeto`):

- **Classe `Pessoa`**: Uma classe simples que implementa `Serializable`. Isso é **necessário** para que os objetos dessa classe possam ser serializados (convertidos para um formato que pode ser escrito em um arquivo).
- **`FileOutputStream` e `ObjectOutputStream`**: Usados para abrir um arquivo e escrever o objeto nele.
- **`writeObject`**: Método que escreve o objeto no arquivo.

Você percebeu que a sintaxe do `try catch` está um pouco diferente do padrão ? (`try {bloco código} catch(Exception e) {bloco código}`).

Nós já vimos isso em uma questão sobre JDBC. É uma característica do Java *chamada **try-with-resources***. Foi introduzida no Java 7 e é usada para garantir que recursos, como streams de I/O, sejam fechados automaticamente após serem usados. Isso ajuda a evitar vazamentos de recursos.

Na sintaxe do *try-with-resources*, você declara os **recursos que quer usar entre parênteses** `()` logo após a palavra-chave `try`. Esses recursos devem implementar a interface `AutoCloseable`, que inclui os métodos `close()`. Quando o bloco `try` termina, quer seja com sucesso ou devido a uma exceção, o método `close()` é chamado automaticamente em cada recurso declarado. Podemos declarar vários recursos perfeitamente em um bloco *try-with-resources*, separando-os com ponto e vírgula.

O `SerialVersionUID` é um número de versão para uma classe serializável. Ele **garante que uma classe pode ser serializada e desserializada corretamente**, mesmo que a classe seja modificada entre o momento da escrita e da leitura.

A saída do console ao executar este código é:

```
Objeto salvo em pessoa.ser
```

Passo 2: Lendo um objeto de um arquivo

```java
package outrostestes;

import java.io.FileInputStream;
import java.io.ObjectInputStream;

public class LeObjeto {
    public static void main(String[] args) {
        try (FileInputStream fileIn = new FileInputStream("pessoa.ser");
             ObjectInputStream in = new ObjectInputStream(fileIn)) {

            Pessoa pessoa = (Pessoa) in.readObject(); // Lê o objeto do arquivo
            System.out.println("Nome: " + pessoa.nome);
            System.out.println("Idade: " + pessoa.idade);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Lendo o Objeto (`LeObjeto`):

- **`FileInputStream` e `ObjectInputStream`**: Usados para abrir um arquivo e ler o objeto dele.
- **`readObject`**: Método que lê o objeto do arquivo e o reconstrói.

A saída do console ao executar este código é:

```
Nome: João

Idade: 30
```