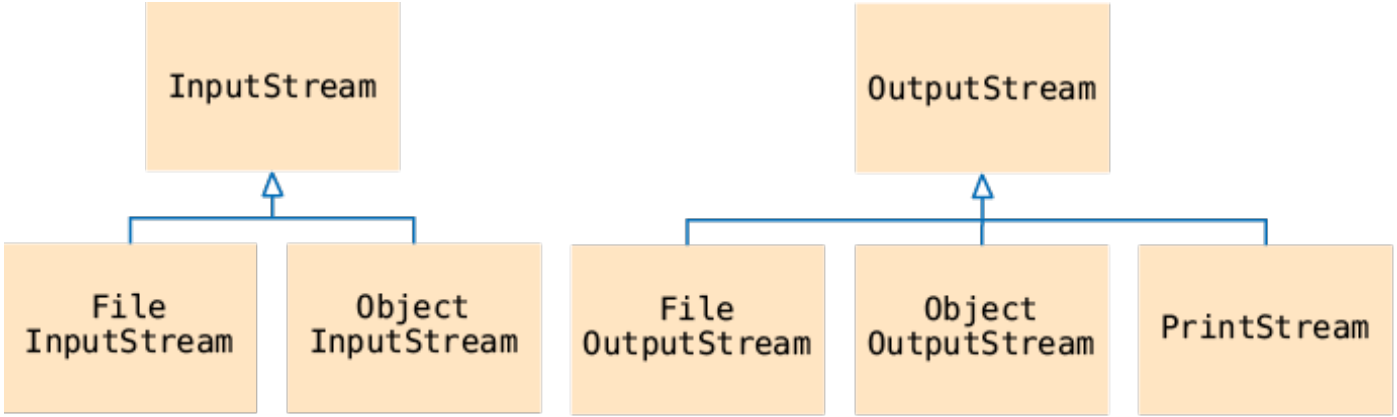
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Leitura e Escrita de Arquivos Binários com Interface Gráfica

Programação Desktop

Prof. Fabrício M. Lopes
fabricio@utfpr.edu.br

Arquivos Binários



O Java fornece dois conjuntos de classes para manipulação de entrada e saída. Os fluxos de entrada e saída tratam de dados binários. A Figura acima exibe uma parte da hierarquia das classes Java para entrada e saída em formato binário.

Classe java.io.FileOutputStream – Construtores

- FileOutputStream(***File*** file)
  - Constrói um objeto FileOutputStream dado um objeto File.
- FileOutputStream(***File*** file, ***boolean*** append)
  - Constrói um objeto FileOutputStream dado um objeto File com um booleano indicando se deve continuar a escrita (true) ou criar um novo arquivo (false).
- FileOutputStream(***String*** fileName)
  - Constrói um objeto FileOutputStream dado um caminho para um arquivo.
- FileOutputStream(***String*** fileName, ***boolean*** append)
  - Constrói um objeto FileOutputStream dado um caminho para um arquivo e um booleano indicando se deve continuar a escrita (true) ou criar um novo arquivo (false).

Classe java.io.FileInputStream

- A classe **FileInputStream** implementa um fluxo de entrada na forma de bytes de um arquivo do sistema de arquivos.
- **FileInputStream** é um meio para leitura de fluxos na forma de bytes, como objetos ou uma imagem por exemplo.
- Construtores:
  - FileInputStream(***File*** file)
    - Cria um objeto FileInputStream, dado o arquivo a ser lido.
  - FileInputStream(***String*** fileName)
    - Cria um objeto FileInputStream, dado o nome de um caminho para um arquivo a ser lido, deve ser um caminho completo.

Classe java.io.ObjectOutputStream

- Um **ObjectOutputStream** escreve objetos java em um fluxo de bytes. Esses objetos podem ser lidos usando um **ObjectInputStream**. Os objetos podem ser armazenados em arquivos binários.
- Apenas objetos cujas classes implemetam a interface **java.io.Serializable** podem ser escritos em fluxos. A classe de cada objeto serializado é codificado usando o **serialVersionUID**.
- O método **writeObject é** usado para escrever um objeto no fluxo de bytes. Os objetos escritos podem ser lidos usando um **ObjectInputstream** com o mesmo tipo e na mesma ordem em que foram escritios.
- Construtor:
  - **ObjectOutputStream**(OutputStream out)
  - Cira um ObjectOutputStream que escreve em um fluxo de saída na forma de bytes especificado pelo parâmetro.

Classe java.io.ObjectInputStream

- A classe **ObjectInputStream** implementa um fluxo de entrada na forma de bytes que foram previamente escritos por um **ObjectOutputStream**.
- As classes **ObjectOutputStream** e **ObjectInputStream** tornam possível o armazenamento persistente de objetos em arquivos no disco, quando utilizados em conjunto com as classes **FileOutputStream** e **FileInputStream** respectivamente.
- Construtor:
  - **ObjectInputStream**(InputStream in)
    - Cria um objeto ObjectInputStream que lê de um fluxo de entrada InputStream definido como parâmetro.

Tratamento de Exceções

- A serialização de objetos em arquivos requer o tratamento de exeções na implementação da escrita e da leitura dos objetos, como definidos a seguir:
  - private void writeObject(java.io.ObjectOutputStream stream)
    - throws IOException;
  - private void readObject(java.io.ObjectInputStream stream)
    - throws IOException, ClassNotFoundException;
  - private void readObjectNoData()
    - throws ObjectStreamException;

Apresentação dos Exemplos

- Cliente.java
- JanelaCliente.java
- ControllerArquivo.java
- ControllerArquivoBinario.java

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.