Serialização de Objetos

Henrique G. G. Pereira
henrique@ufsm.br

Nas aulas anteriores

- Orientação a Objetos
  - Classes
  - Atributos
  - Métodos
- Arquivos
  - Leitura
  - Escrita

Introdução

- Problema:
  - Você desenvolveu um objeto complexo:
    - Tabuleiro de um jogo
    - Tabela com dados estatísticos
    - Armazém de dados
    - Etc.
  - Você não quer que esses objetos sejam destruídos ao fechar o programa.
    - Você gostaria de manter os mesmos estados
  - Você gostaria de compartilhar esses objetos com outros programas.

Introdução (cont.)

- Você quer que o objeto continue existindo.
  - Mesmos métodos
  - Mesmos estados
  - Mesmas variáveis
  - Mesmos valores.

- Você quer que o objeto tenha ***persistência***.

O que é persistência?

- Persistência de Objetos
  - Durabilidade

  - Capacidade de existir além do programa onde foi criado.
    - Tempo
      - Ao ser possível recarregar uma fase.
      - Ao ser possível recarregar todos os valores preenchidos anteriormente.
    - Espaço
      - Ao ser possível reutilizar o objeto ou parte dele em outro programa.

Alcançando a persistência

- Implementando métodos próprios
  - Salvar
  - Recarregar

- Utilizando um protocolo de **serialização de objetos**.

Serialização de Objetos

- Técnica que permite a persistência de objetos
  - Permite também a passagem de objetos para serviços remotos.
- Define uma maneira para exportar os valores internos de um objeto.
  - Em uma sequência de bytes.
  - ou Utilizando uma notação específica.
    - JSON, BSON.

Como funciona a serialização?

- Identificação das variáveis internas de um objeto.
- Transformação dessas variáveis em tipos primitivos.
  - Números, Strings, etc.
- Escrita desses valores em um ***byte stream***.

Como funciona a serialização? (cont.)

- Funciona para objetos dentro de objetos
  - Se um objeto faz referência a outro objeto e esse objeto faz referência a outro objeto
    - Todos são salvos juntos
- O conjunto de objetos relacionados é chamado de Grafo de Objetos
  - A serialização converte esse grafo inteiro para bytes.

Exercícios para fixação teórica

1. Em que cenários você poderia implementar a serialização?

2. Utilizando suas palavras, defina o que é a persistência de um objeto e cite um dos seus objetivos.

Serialização de Objetos em Java

- Requer a implementação da Interface *java.io.Serializable*
  - *adicionar essa interface a lista de implementações.*
  - *Definir um método construtor que não receba parâmetros.*

- *Permite a serialização*
  - ObjetOutputStream

- *Permite a deserialização*
  - ObjectInputStream

  - *Processo inverso*
    - *Criação de um objeto a partir de um stream de bytes.*

Requisitos para a serialização

- Todos os objetos referenciados dentro de um objeto precisam ser serializáveis
  - Devem implementar java.io.Serializable
    - E todos os objetos dentro desses objetos também.
      - E assim por diante.

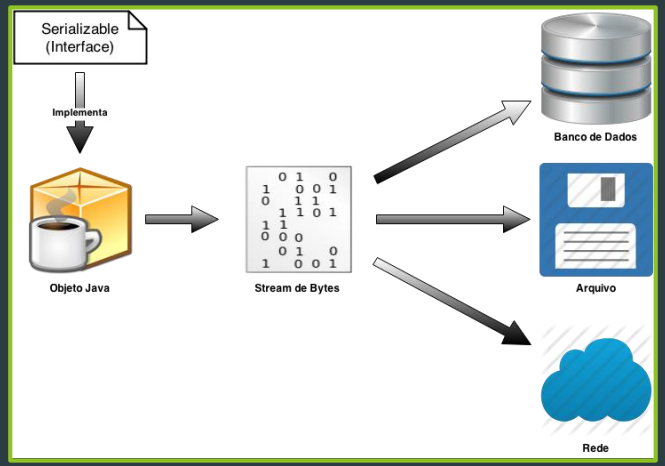
Exemplo da classe Pessoa

```
public class Pessoa implements java.io.Serializable {
  public String nome;
  private String cpf;
  private String senha;

  public class Pessoa(){
  }

  ...
}
```

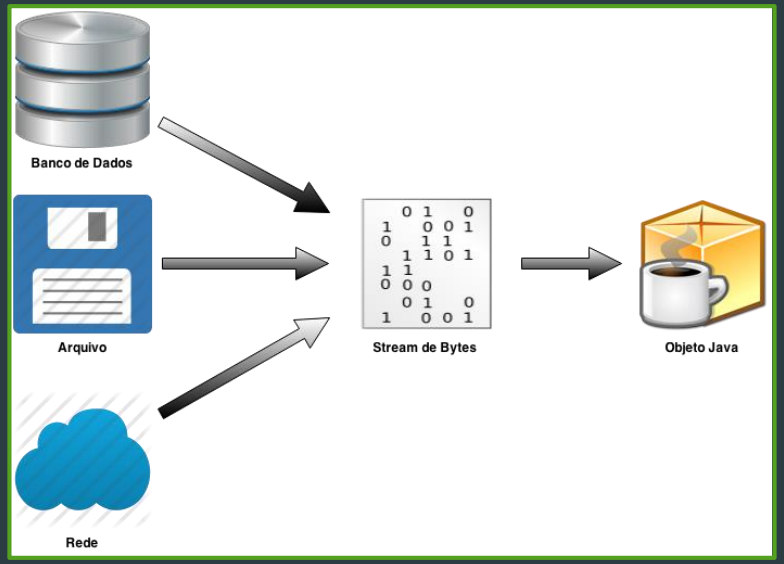
Processo de Serialização



Serialização para arquivo

```
FileOutputStream arquivo = new FileOutputStream("data.ser");
ObjectOutputStream saida = new ObjectOutputStream(arquivo);
saida.writeObject(pessoa);
saida.close();
```

Processo de Deserialização



Deserialização a partir de arquivo

```
FileInputStream arquivo = new FileInputStream("data.ser");
ObjectInputStream entrada = new ObjectInputStream(arquivo);
Pessoa p = (Pessoa) entrada.readObject();
entrada.close();
```

Valores não serializáveis

- Existem situações onde é possível não se deseja serializar todos dados
  - Ex.: Dados temporários
    - Idade calculada com base na data de nascimento
  - Ex.: Dados sensíveis e que não terão utilidade em objetos remotos
    - Senhas
- Para essas situações existe o modificador ***transient***
  - Indica que o dado é transitório
    - Não deve ser armazenado
    - Não faz sentido armazenar ele

Campo cpf com *transient*

```
public class Pessoa implements java.io.Serializable {
  public String nome;
  private String cpf;
  private transient String senha;

  public class Pessoa(){
  }

  ...
}
```

Exceções na serialização

- Alguns campos não podem ser serializados
  - Campos estáticos
    - Definidos como ***static***
    - Pertencem a **Classe** e não a instância do objeto

Problemas da serialização

- O que acontece se ocorrer alguma alteração na classe de um objeto serializado?
  - PROBLEMA!
    - Se o bytecode da classe for diferente
    - Se a classe mudar depois que o objeto for serializado
  - *InvalidClassException*

Detalhes de Implementação

- A informação da classe de origem também é armazenada no objeto serializado.
  - Atributo ***serialVersionUID***.
    - Computado automaticamente.
    - poder ser sobrescrito.
      - Não é uma boa idéia.

Exercício para a próxima aula

1. Implemente uma classe Família que deverá ser serializável. Uma Família possui um sobrenome e uma lista de Pessoas que pertencem aquela família.

Obs. Não esqueça de que todos os membros de uma classe serializável precisam permitir a serialização.

Resumo de Hoje

- Conceitos básicos sobre persistência
- Serialização e deserialização de objetos
  - Exemplos de uso
  - Problemas e Exceções

Próxima Aula

- Interagindo com objetos remotos
  - *Remote Method Invocation*
    - RMI

Referências

- FURGERI, S. **Java 7: Ensino didático**. 2ª edição. Editora Érica, São Paulo, 2012.
- DEITEL, P. **Java – Como programar**. 8a edição. Editora Prentice Hall, 2010.
- HORSTMANN, C. **Core Java.** 7ª edição. Editora Prentice Hall, 2009.