



Tópicos avançados em Programação

Serialização e a manipulação de arquivos

<http://dl.dropbox.com/u/3025380/prog2/aula6.pdf>

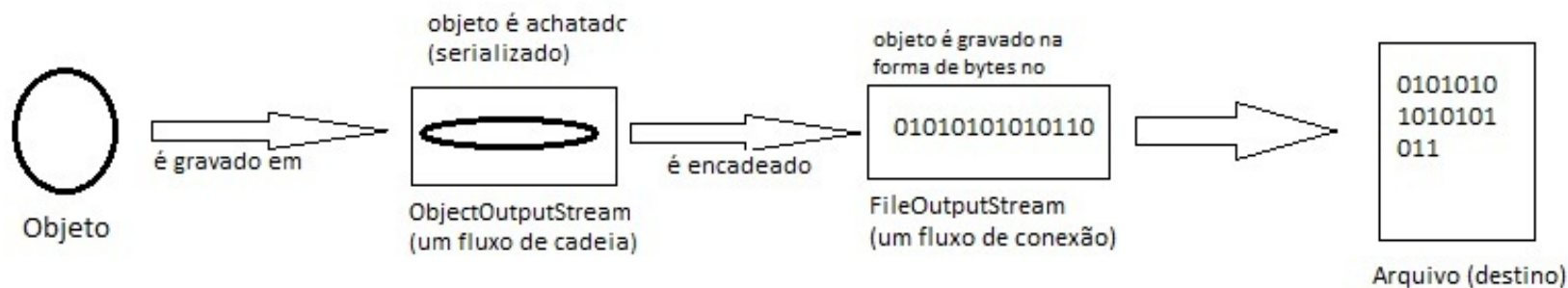
flavio.cec@unisul.br

Serialização

- O que é serialização?
 - A serialização resume-se em um único objetivo salvar, gravar e capturar o estado de um objeto.
 - Ou seja, tenho um objeto de uma classe e quero salvar seu estado, então uso serialização, pois quando quiser usar esse objeto com seu último estado gravado é somente deserializá-lo.

Serialização

- Mas por que usar serialização?
 - A partir do momento que se serializa um objeto ou uma variável de instância, seu valor é gravado (em um arquivo), armazenado e futuramente posso obter esse valor através da deserialização.



Fonte: <http://bit.ly/14qk1BF>

Serialização

- Serialização em Java:
 - Para que um objeto seja serializado em Java é necessário que o mesmo implemente a interface ***Serializable***.

```
import java.io.Serializable;

public class Cliente implements Serializable {

    private static final long serialVersionUID = 6161029930731563853L;

    private int matricula;

    private String nome;

    private int telefone;

    public int getMatricula() {
        return matricula;
    }
}
```



Sempre que se implementa a interface ***Serializable*** é necessário gerar esse atributo.

Serialização

- Preparando uma lista de clientes para serializar:

```
Cliente cliente = null;  
List<Cliente> clientes = new ArrayList<Cliente>();
```

```
cliente = new Cliente();  
cliente.setMatricula(123);  
cliente.setNome("Cliente 1");  
clientes.add(cliente);
```

```
cliente = new Cliente();  
cliente.setMatricula(321);  
cliente.setNome("Cliente 2");  
clientes.add(cliente);
```

```
this.serializandoClientes(clientes);
```

→ Método criado para serializar o objeto.

Serializando a lista de clientes

```
private void serializandoClientes(List<Cliente> clientes) {
```

```
    OutputStream escritorByte = null;
```

```
    ObjectOutputStream escritorObjeto = null;
```

```
    try {
```

```
        escritorByte = new FileOutputStream("conteudo/clientes.bin", true);
```

```
        escritorObjeto = new ObjectOutputStream(escritorByte);
```

```
        escritorObjeto.writeObject(clientes);
```

```
        escritorObjeto.flush();
```

```
    } catch (FileNotFoundException e) {
```

```
        System.err.println(e);
```

```
    } catch (IOException e) {
```

```
        System.err.println(e);
```

```
    } finally {
```

```
        try {
```

```
            if(escritorObjeto != null) {
```

```
                escritorObjeto.close();
```

```
            }
```

```
            if(escritorByte != null) {
```

```
                escritorByte.close();
```

```
            }
```

```
        } catch (Exception e){}
```

```
    }
```

```
}
```

A classe **ObjectOutputStream** é responsável pela escrita do objeto

Deserializando a lista de clientes

```
@SuppressWarnings("unchecked")
public void deserializarCliente() {
    InputStream leitorByte = null;
    ObjectInputStream leitorObjeto = null;

    try {

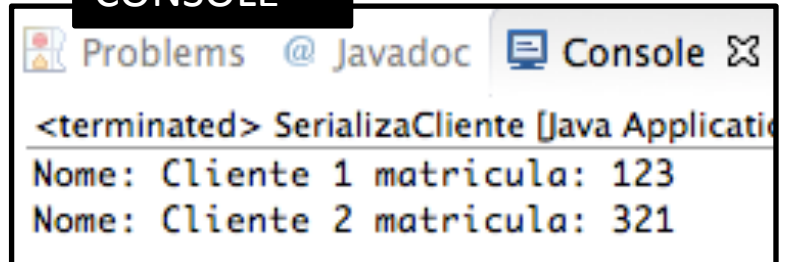
        leitorByte = new FileInputStream("conteudo/clientes.bin");
        leitorObjeto = new ObjectInputStream(leitorByte);

        List<Cliente> clientes = (List<Cliente>) leitorObjeto.readObject();

        for(Cliente cliente : clientes) {
            System.out.println("Nome: "+cliente.getNome()+" matricula: "+cliente.getMatricula());
        }

    } catch (FileNotFoundException e) {
        System.err.println(e);
    } catch (IOException e) {
        System.err.println(e);
    } catch (ClassNotFoundException e) {
        System.err.println(e);
    } finally {
        try {
            if(leitorObjeto != null) {
                leitorObjeto.close();
            }
            if(leitorByte != null) {
                leitorByte.close();
            }
        }
    } catch (Exception e){}
}
```

CONSOLE



The screenshot shows an IDE console window with tabs for Problems, Javadoc, and Console. The Console tab is active, displaying the output of the Java application. The output consists of two lines: "Nome: Cliente 1 matricula: 123" and "Nome: Cliente 2 matricula: 321".

```
<terminated> SerializaCliente [Java Applicati
Nome: Cliente 1 matricula: 123
Nome: Cliente 2 matricula: 321
```

Continuando o Trabalho
Integrador...

Trabalho Integrador

- Lembram do cenário da biblioteca universitária da Unisul?
- Surgiu um novo requisito onde foi solicitado que todas as informações sejam persistidas em arquivos (serializados).
 - Dessa forma os dados devem ficar salvo mesmo depois da finalização do sistema.

Trabalho Integrador

- Requisitos:
 - O sistema deve armazenar todas informações em memória de modo que seja utilizada a estrutura de dados mais adequada para as operações em questão;
 - O sistema deve permitir o cadastro de alunos;
 - O sistema deve permitir o cadastro de professores
 - Tanto alunos como professores devem ser mantidos na mesma estrutura;
 - A forma de consulta de ambos é via sua matrícula;

Trabalho Integrador

- Requisitos:
 - O sistema deve permitir o cadastro de livros;
 - Não são permitidos o cadastro de livros repetidos;
 - Um livro deve ter uma lista de exemplares;
 - O sistema deve permitir o vínculo de um exemplar com um usuário (aluno ou professor) da biblioteca
 - Cada usuário pode ter até 5 livros emprestados simultaneamente (não é permitido pegar mais de um exemplar do mesmo livro)

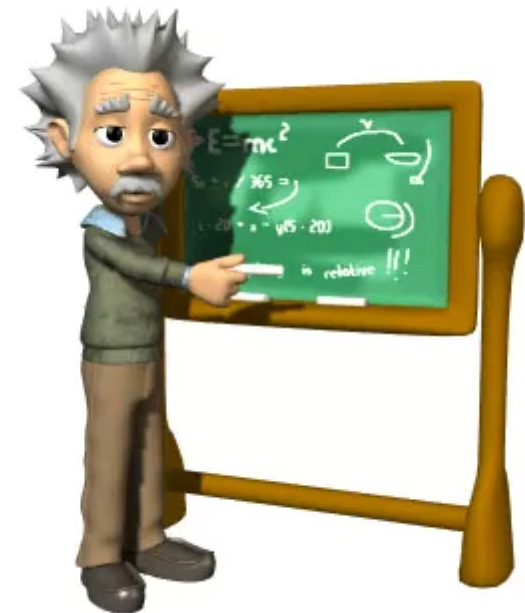
Trabalho Integrador

- Característica do aluno:
 - Matricula;
 - Nome;
 - Nome do curso;
 - Exemplares pegos;



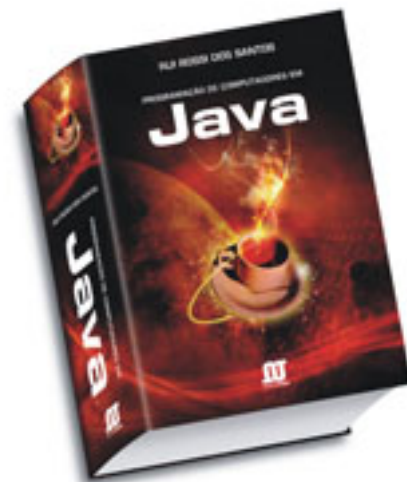
Trabalho Integrador

- Característica do professor:
 - Matricula;
 - Nome;
 - Lista de cursos que participa;
 - Exemplares pegos;



Trabalho Integrador

- Característica do livro:
 - Título;
 - Autor;
 - Lista de exemplares;



Trabalho Integrador

- Característica do exemplar:
 - Código;
 - Localização
 - Edição



Trabalho Integrador

- O sistema deve possuir uma interface (menu) para navegar entre as opções de operações;
- Outras operações permitidas (além do cadastro):
 - Fazer uma consulta por livro e/ou exemplar;
 - A partir da visualização da consulta o usuário pode fazer as seguintes operações:
 - Alterar os dados do objeto;
 - Excluir o objeto da estrutura de dados.

IMPORTANTE

- A partir da aula 8, utilizaremos o banco de dados (SGBD) PostgreSQL.
- É importante que a sua instalação seja feita anterior a data da nossa próxima aula.
- Tutorial para instalação e configuração do Banco:
 - Windows:
https://dl.dropbox.com/u/3025380/BD2/postgres_windows.pdf
 - MacOS:
https://dl.dropbox.com/u/3025380/BD2/postgres_mac.pdf
 - Linux (Ubuntu):
https://dl.dropbox.com/u/3025380/BD2/postgres_linux.pdf

Próxima aula prova

- Conteúdo:
 - Aula 1;
 - Aula 2;
 - Aula 3;
 - Aula 4;
 - Aula 5;
 - Aula 6.
- Importante:
 - Refazer os exercícios!

