

# ALGORITMOS E ESTRUTURAS DE DADOS 2023/2024 SERIALIZAÇÃO EM JAVA

---

Armanda Rodrigues

4 de outubro de 2023

# Serialização em Java

- O Interface Serializable faz parte do Pacote java.io
- Tipos Primitivos em Java – São todos serializáveis
- Vetores e Strings são serializáveis
- Relativamente aos TADS por nós criados, estes devem estender o interface Serializable
  - As instâncias das classes que sejam implementações dos TADs serializáveis, são também serializáveis
- As variáveis de classe (static), não são serializáveis

# Classes serializáveis

- Devem declarar uma constante (de classe) do tipo `long`, chamada `serialVersionUID`, com um valor qualquer.

```
static final long serialVersionUID = 0L;
```

- Sem esta constante, o compilador dá um warning.

# Classes para a Serialização

- ObjectOutputStream – Classe que permite a serialização do estado de um objeto, para um ficheiro
- ObjectInputStream – Classe que permite a leitura do objecto serializado para a memória do programa

# ObjectOutputStream

- Pacote: `java.io`;
- **Construtor:**
  - `ObjectOutputStream (OutputStream out) throws IOException`
- **Métodos:**
  - `void writeObject( Object obj ) throws IOException`
  - `void writeBoolean( boolean val ) throws IOException`
  - `void writeInt( int val ) throws IOException`
  - `void flush( ) throws IOException`
  - `void close( ) throws IOException`

# ObjectInputStream

- Pacote: `java.io`;
- **Construtor:**
  - `ObjectInputStream( InputStream in ) throws IOException`
- **Métodos:**
  - `Object readObject( ) throws IOException, ClassNotFoundException`
  - `boolean readBoolean( ) throws IOException`
  - `public int readInt( ) throws IOException`
  - `public void close( ) throws IOException`

# Exemplo – Interface Letter

```
import java.io.Serializable;

public interface Letter extends Serializable{

    .....

}
```

# Exemplo – Classe LetterClass

```
public class LetterClass implements Letter{

    static final long serialVersionUID = 0L;

    // Address implements Serializable.
    private Address returnAddress;

    // Date implements Serializable.
    private Date date;
    private String opening, closing;

    // LetterBody implements Serializable.
    private LetterBody body;

    public LetterClass( ... ) { ...}
        .....
}
```



## Exemplo – Classe MyLetterIO (1)

```
import java.io.*;
public class MyLetterIO{

    private Letter letter;
    private String fileName;

    public MyLetterIO( ..., String theFileName ){
        letter = new LetterClass( ... );
        fileName = theFileName;
    }
    .....

    public void load( ) { ...}

    public void store( ) { ...}
}
```

Isto pode não ser uma boa ideia

## Exemplo – Classe MyLetterIO (2)

```
public void store( ){  
    try{  
        ObjectOutputStream file = new ObjectOutputStream(  
            new FileOutputStream(fileName) );  
        file.writeObject(letter);  
        file.flush();  
        file.close();  
    }  
    catch ( IOException e )  
    { ... }  
}
```

NÃO DEVE SER VAZIO!  
Inclua sempre aqui uma forma  
de saberem que houve um erro

## Exemplo – Classe MyLetterIO (3)

```
public void load( ){  
    try{  
        ObjectInputStream file = new ObjectInputStream(  
            new FileInputStream(fileName) );  
  
        // Compiler gives a warning.  
        letter = (Letter) file.readObject();  
        file.close();  
    }  
    catch ( IOException e )  
    { ... }  
  
    catch ( ClassNotFoundException e )  
    { ... }  
}
```

NÃO DEVE SER VAZIO!  
Inclua sempre aqui uma forma  
de saberem que houve um erro

# Outro exemplo – Bank

```
public static void main(String[] args) throws
    FileNotFoundException, IOException, ClassNotFoundException {
```

```
    BankSys bank = load();
```

Verifica existência do ficheiro e devolve objecto

```
    Scanner in = new Scanner(System.in);
    String cmd = in.next().toUpperCase();
    while (!cmd.equals(EXIT)) {
        switch (cmd) {
            case ADD_ACC: addAcc(in, bank); break;
            case SEARCH_ACC: searchAcc(in, bank); break;
            ...
            default: break;
        }
        System.out.println();
        cmd = in.next().toUpperCase();
    }
```

```
    store(bank);
```

load e save estão fora do ciclo...

```
}
```

## Outro exemplo – Bank (save)

é o store da classe

```
private static void save(BankSys bank) {  
    try {  
        ObjectOutputStream oos =  
            new ObjectOutputStream(new FileOutputStream(BANK_FILE));  
        oos.writeObject(bank);  
        oos.close();  
    }  
    catch (IOException e) {  
        //Para testar localmente, remover quando submeter no Mooshak  
        //  
        System.out.println("Issue in writing...");  
    }  
}
```

Por alguma razão não se conseguiu escrever o ficheiro – Não deverá ser relevante no Mooshak

# Outro exemplo – Bank (load)

```

@SuppressWarnings("unchecked")
private static BankSys load() {
    try {
        ObjectInputStream ois =
            new ObjectInputStream(new FileInputStream(BANK_FILE));
        BankSys bank = (BankSys) ois.readObject();
        ois.close();
        return bank;
    }
    catch (IOException e) {
        // Ficheiro não existe: Criar um objeto vazio
        // Mensagem para usar localmente, remover na submissão ao Mooshak
        System.out.println
            ("Non existing serialization file: Creating new Object.");
        return new BankSysClass();
    }
    catch (ClassNotFoundException e) {
        // Problemas na serialização das classes associadas ao objeto: verificar todos os interfaces e classes
        // Mensagem para usar localmente, remover na submissão ao Mooshak
        System.out.println
            ("Problems with serialization: Creating new Object.");

        return new BankSysClass(); }
}

```

Devolve objeto do tipo **BankSys**

Situação normal na primeira execução do Programa: Não existe ainda ficheiro de serialização

Esta situação não é normal: Há interfaces/classes no objeto não serializáveis

# Pacote dataStructures

- Todos os Interfaces disponíveis na página de AED estendem Serializable
  - Stack, Queue, List, Dictionary, Iterator, Entry
- Classes que implementam Serializable
  - DoubleListNode<E>