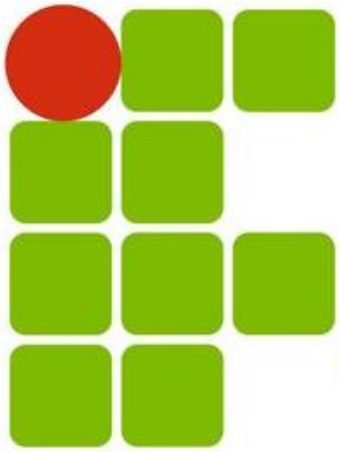



# POO2: Streams e Serialização

Felipe Frechiani de Oliveira

felipefo@gmail.com

# Agenda

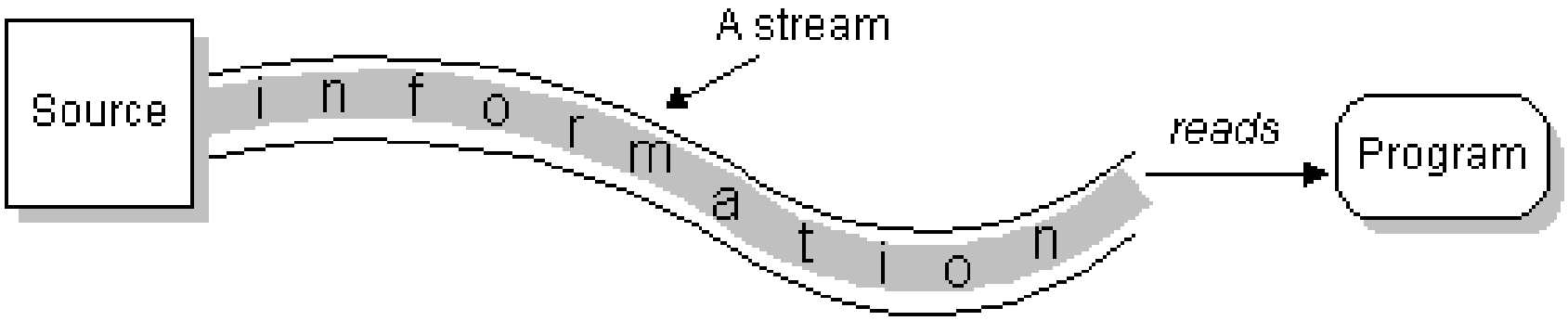
- Streams
- Serialização

# Stream

A **stream** is a sequence of data elements made available over time

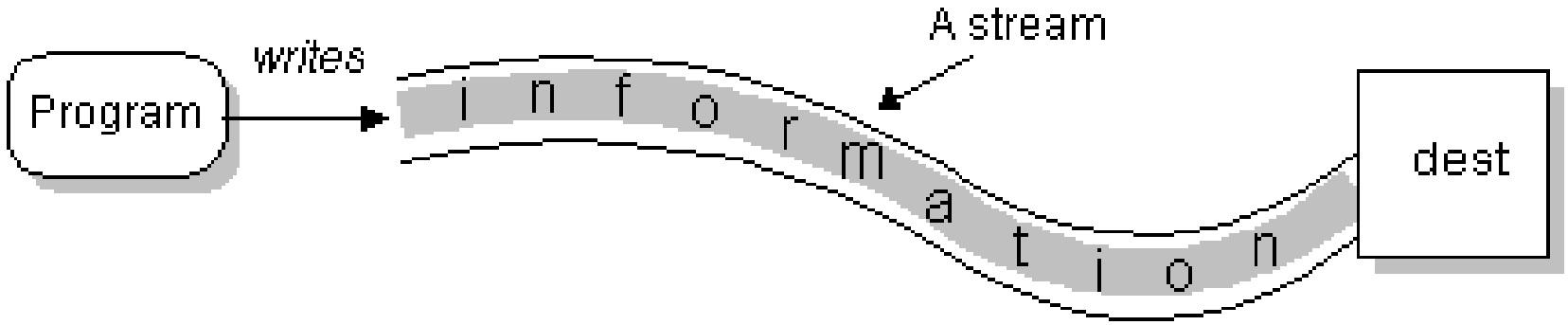
# Stream

O programa abre uma stream através de uma fonte de informação( arquivo, memória, socket) e lê essa informação de forma sequencial.



# Stream

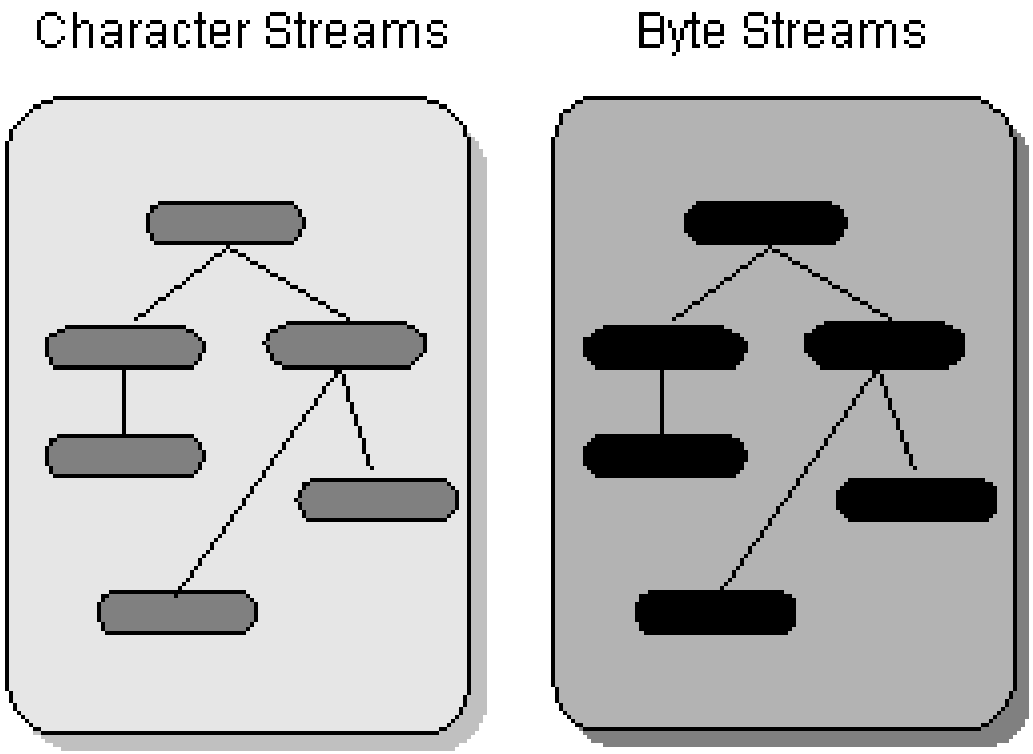
Dá mesma forma um programa pode enviar para um destinatário externo. Esse processo se dá através da abertura de um *stream* para um destinatário e consequentemente escrita de forma sequencial.



# Read and Writing

| Read | Writing |
| --- | --- |
| open a stream | open a stream |
| while more information | while more information |
| read information | write information |
| close the stream | close the stream |

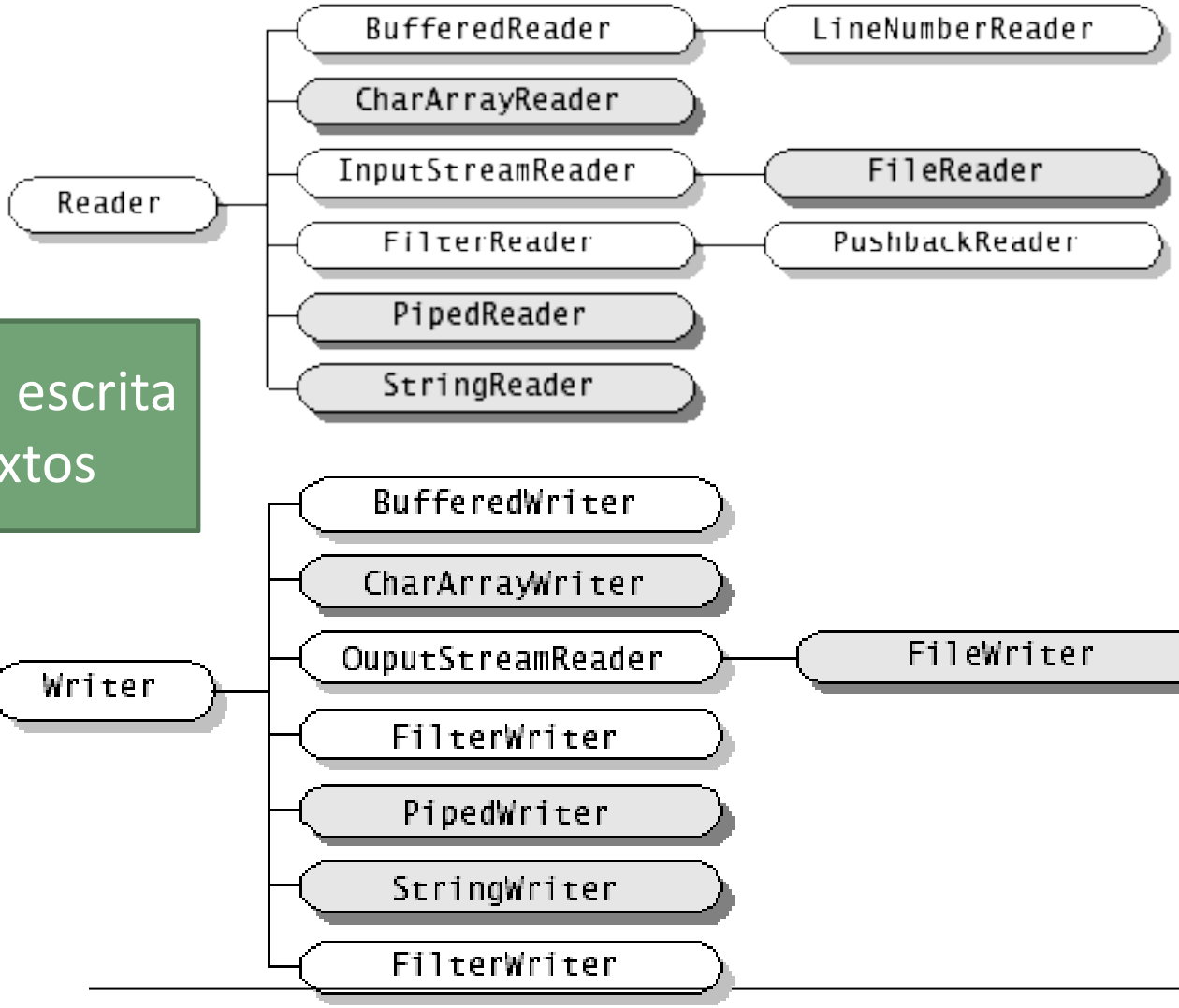
# Char and Byte Streams




Manipular caracteres

Manipulador de propósito geral

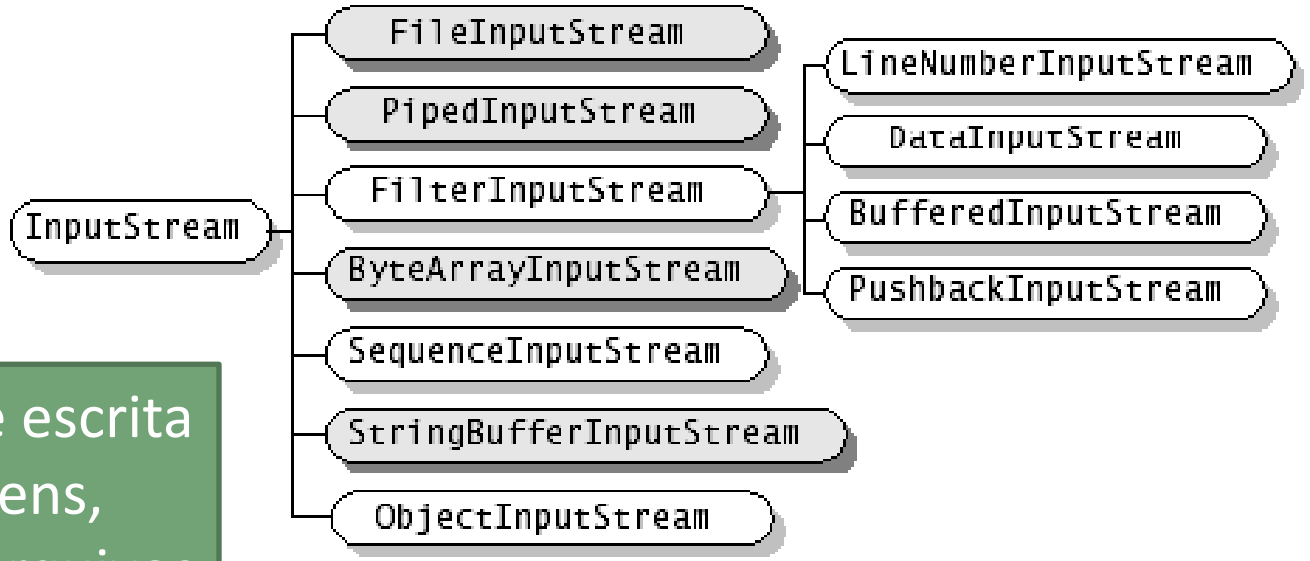
# Char stream



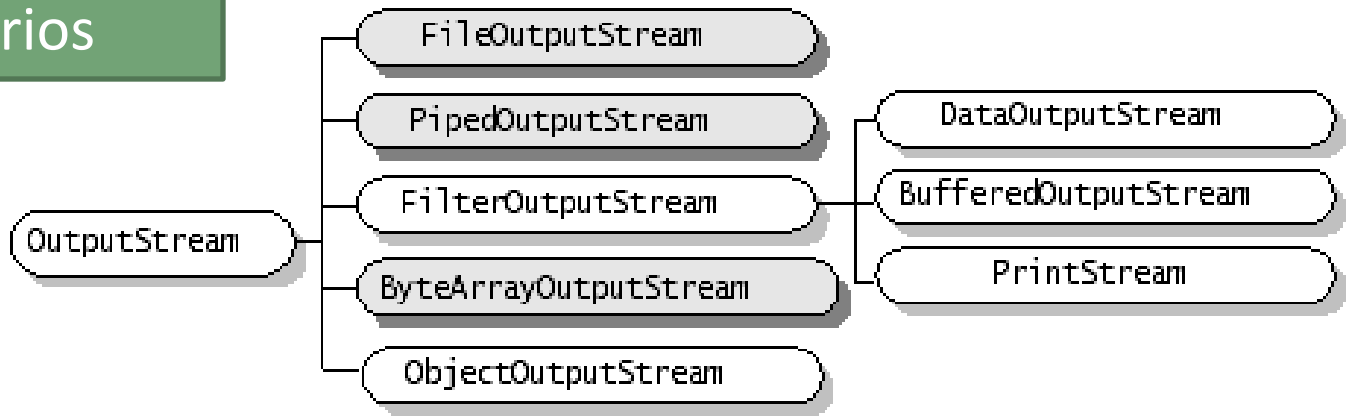

Leitura e escrita de textos

INSTITUTO FEDERAL
ESPIRITO SANTO

# Byte Stream




Leitura e escrita imagens, vídeos, arquivos binários




INSTITUTO FEDERAL
ESPIRITO SANTO

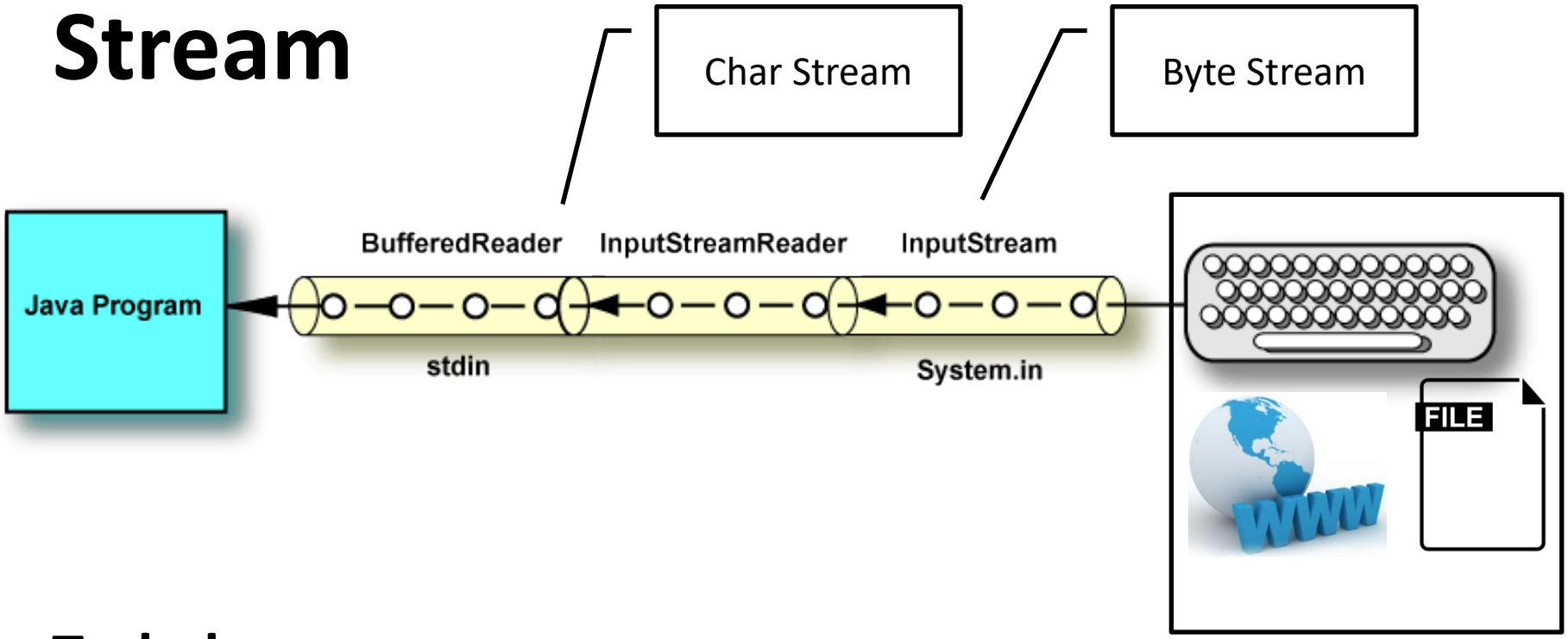
# Char Stream e Byte Stream

Reader e InputStream definem funções similares, porém para tipos de dados diferentes.

- **Reader** contém métodos para leitura de caracteres e arrays de caracteres;

- **InputStream** define os mesmos métodos porém para ler bytes e array de bytes.

# Stream




**Teclado**:

BufferedReader stdin = new BufferedReader( new InputStreamReader( System.in ) );

# Byte Stream vs Char Stream

```
InputStreamReader inputStream = new  InputStreamReader( System.in );
System.out.print("Enter a line:");
String input= "";
int data = inputStream.read();
input +=  (char)(data);
   while ((char)(data) != '\n') {
      data = inputStream.read();
      char caracter = (char)(data);
      input += caracter;
}
System.out.print(input);
inputStream.close();
```

Byte Stream

```
BufferedReader stdin = new BufferedReader(
          new  InputStreamReader( System.in ) );

System.out.print("Enter a line:");
System.out.println(stdin.readLine());
stdin.close();
```

Char Stream

INSTITUTO FEDERAL
ESPIRITO SANTO

# File Stream - ArquivoFileStream.java

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

    String current = new java.io.File( "." ).getCanonicalPath();
    File inputFile = new File(current + "\\src\\arquivo_entrada.txt");
    File outputFile = new File(current + "\\src\\arquivo_saida.txt");
    FileReader in = new FileReader(inputFile);
    FileWriter out = new FileWriter(outputFile);
    int c;
    while ((c = in.read()) != -1) {
        out.write(c);
    }

    in.close();
    out.close();

    System.out.println("Arquivo gerado em: " + outputFile.getAbsolutePath());
}
```

# Exercício

Exercício ManipulacaoCV

# Serialização de Objetos

# Object Serialization

1. Escreve os metadados associados ao objeto. Exemplo nome da classe, nomes dos atributos, etc;

2. Escreve de forma recursiva a descrição das superclasses até encontrar o java.lang.Object;

3. Quando ele termina de escrever as informações de metadados ele começa a escrever as informações associadas a instância do objeto.

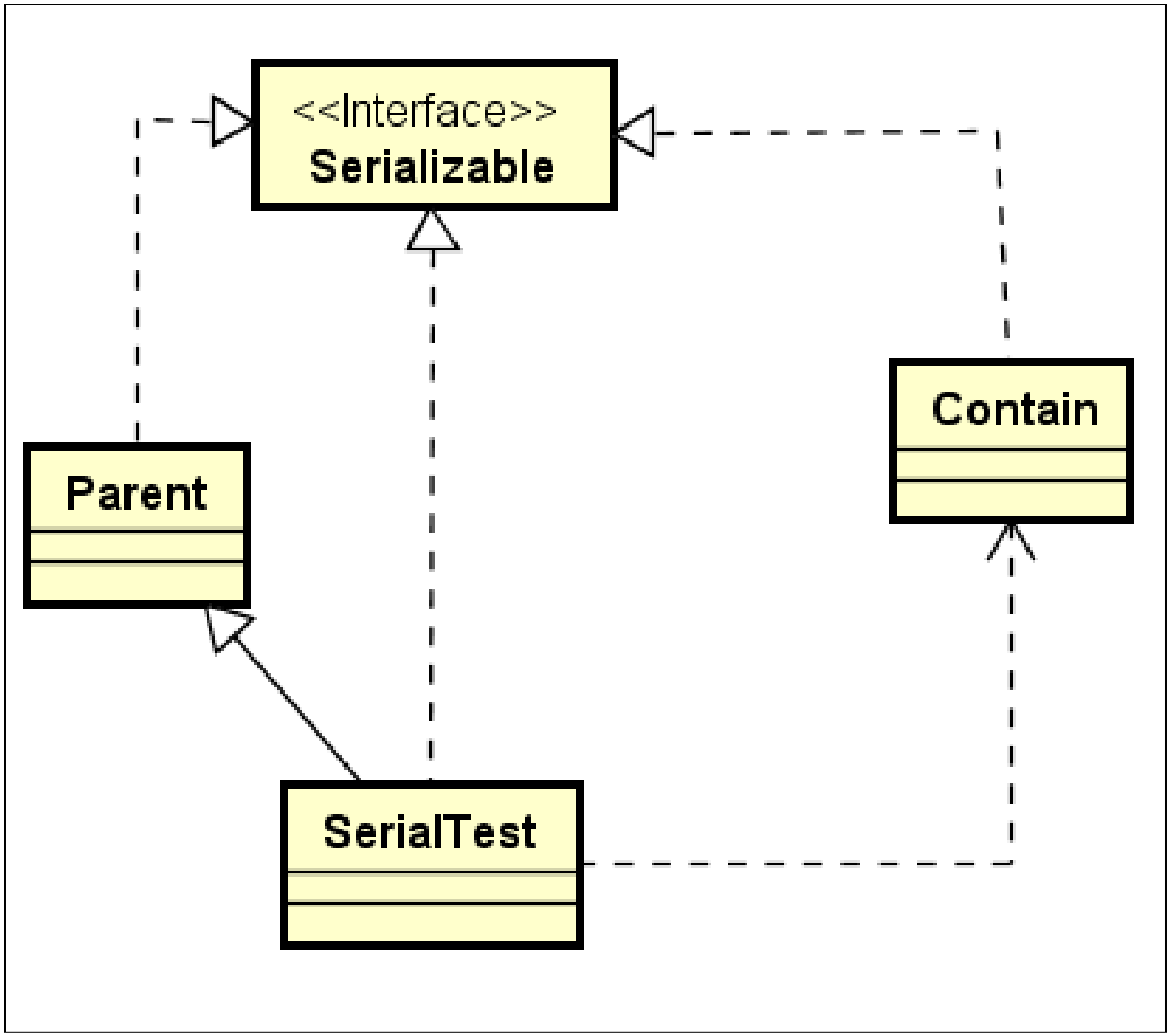
```
class parent implements Serializable {
        int parentVersion = 10;
}

class contain implements Serializable{
        int containVersion = 11;
}
public class SerialTest extends parent implements Serializable {
        int version = 66;
        contain con = new contain();

        public int getVersion() {
                return version;
        }
        public static void main(String args[]) throws IOException {
                FileOutputStream fos = new FileOutputStream("temp.out");
                ObjectOutputStream oos = new ObjectOutputStream(fos);
                SerialTest st = new SerialTest();
                oos.writeObject(st);
                oos.flush();
                oos.close();
        }
}
```

Nome do arquivo que vai conter o objeto serializado

Salvando objeto serializado


<<Interface>>
Serializable
Parent
Contain
SerialTest

# SerialTest - Serializado

AC ED: STREAM_MAGIC. Specifies that this is a serialization.

53 65 72 69 61 6c 54 65 73 74: SerialTest, the name of the class.

00 05: STREAM_VERSION. The serialization version.

```
AC ED 00 05 73 72 00 0A 53 65 72 69 61 6C 54 65
73 74 05 52 81 5A AC 66 02 F6 02 00 02 49 00 07
76 65 72 73 69 6F 6E 4C 00 03 63 6F 6E 74 00 09
4C 63 6F 6E 74 [illegible] 70 61 72
65 6E 74 0E DB [illegible] 01 49 00
0D 70 61 72 65 [illegible] 6E 78 70
00 00 00 0A 00 00 00 42 73 72 00 07 63 6F 6E 74
61 69 6E FC BB E6 0E FB CB 60 C7 02 00 01 49 00
0E 63 6F 6E 74 61 69 6E 56 65 72 73 69 6F 6E 78
70 00 00 00 0B
```

# Object Serialization

```
UsuarioSerializado usuarioSerializado = new UsuarioSerializado();
usuarioSerializado.setEmail("joao@email.com");
usuarioSerializado.setNome("joao");
usuarioSerializado.setSenha("password");

FileOutputStream fileSerializado = new FileOutputStream(
        current + "\\src\\usuario_serializado.ser");
ObjectOutputStream outputSerializado = new ObjectOutputStream(fileSerializado);
outputSerializado.writeObject(usuarioSerializado);
outputSerializado.close();

System.out.println("Usuario Salvo");
System.out.println("Arquivo gerado em: " + current +
        "\\src\\usuario_serializado.ser");
```

```
write(byte[] bytes)                    void
write(int i)                           void
write(byte[] bytes, int i, int i1)     void
```

# Object DeSerialization

```
String current = new java.io.File(".").getCanonicalPath();

FileInputStream fileIn = new FileInputStream(current +
        "\\src\\usuario_serializado.ser");
ObjectInputStream in = new ObjectInputStream(fileIn);
user = (UsuarioSerializado) in.readObject();
in.close();

System.out.println("Deserializando Usuario...");
System.out.println("Nome: " + user.getNome());
System.out.println("Email: " + user.getEmail());
System.out.println("Senha: " + user.getSenha());
```

# Exercício

Exercício Serialização Mensagem(moodle)