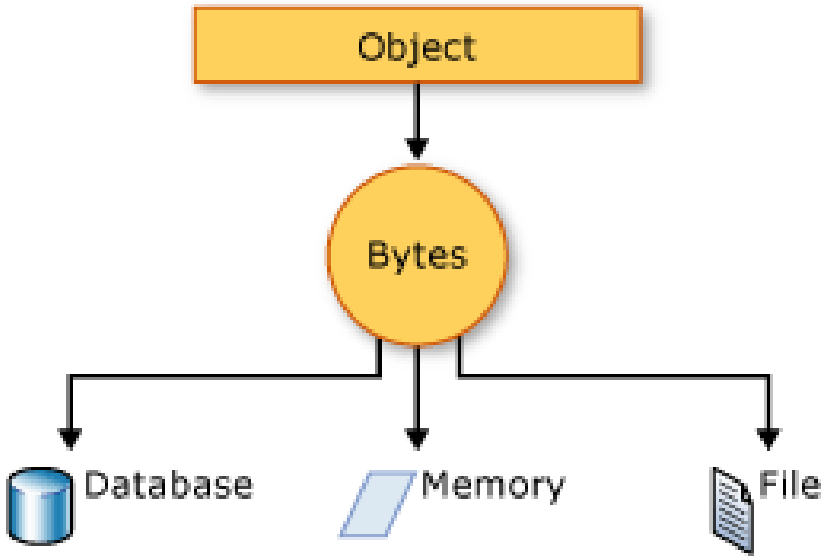
Serialização de Objetos



Serialização é o processo de conversão de um objeto em um fluxo de bytes para armazenar o objeto ou fluxo na memória, em um banco de dados, ou em um arquivo. Sua finalidade principal é salvar o estado de um objeto para ser capaz de recriá-lo quando necessário. O processo inverso é chamado desserialização.

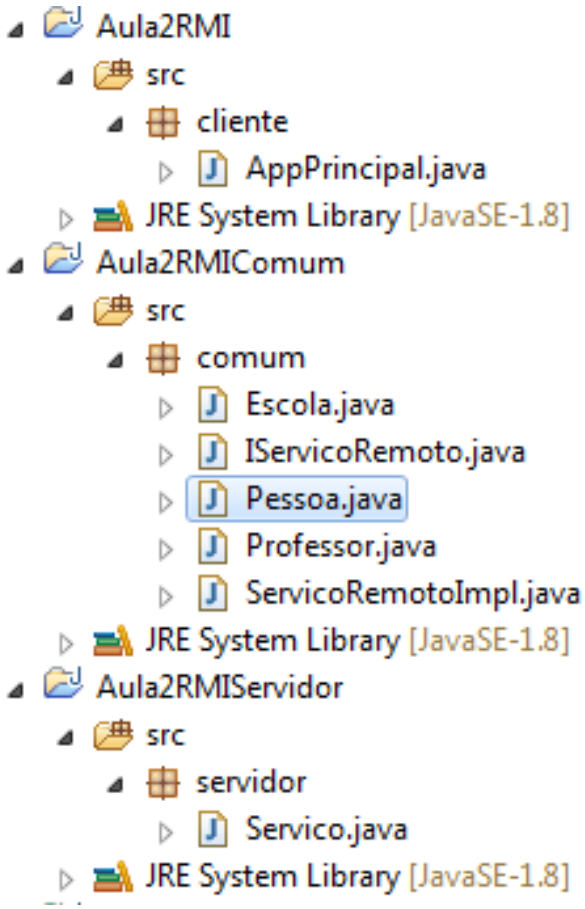
- Stream Java transmitem cadeias de bytes
- Java possibilita que objetos serializáveis sejam transmitidos via streams
- Um objeto é serializável se implementa a interface Serializable
- Na serialização os atributos de um objeto são convertidos numa sequência de bytes e na desserialização o objeto é recuperado a partir da sequência de bytes.

RMI (Remote Method Invocation)

- Tecnologia para possibilitar a invocação de método remoto como se fosse método local.
- Possibilita a comunicação entre objetos em diferentes máquinas virtuais.
- Serviços providos por objetos remotos são descritos por interfaces codificadas na própria linguagem Java.
- Modelo de execução: cliente-servidor
- Embora a invocação de métodos remotos seja executada da mesma forma que para métodos locais, a invocação de um método remoto pode falhar.
- Invocações de métodos remotos obrigatoriamente devem ser tratadas através de exceções.

Exemplo prático

Iremos criar 3 projetos:



Aula2RMIComum

Classe Escola:

```
1  package comum;
2
3  import java.rmi.RemoteException;
4
5  public class Escola extends Pessoa {
6
7      public Escola() throws RemoteException {
8          super();
9      }
10
11     private static final long serialVersionUID = -1905487820086139957L;
12
13 }
14
```

Classe Professor:

```java
package comum;

import java.rmi.RemoteException;

public class Professor extends Pessoa{

    public Professor() throws RemoteException {
        super();
    }
    private static final long serialVersionUID = -2400835714357147362L;
}

```

Classe Pessoa:

```java
package comum;

import java.io.Serializable;

public class Pessoa implements Serializable{

    private static final long serialVersionUID = -3494965244430818150L;

    private int id;
    private String nome;
    private String telefone;
    private String cidade;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getTelefone() {
        return telefone;
    }
    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
    public String getCidade() {
        return cidade;
    }
    public void setCidade(String cidade) {
        this.cidade = cidade;
    }
}

```

Interface IServicoRemoto:

```java
package comum;

import java.rmi.Remote;

public interface IServicoRemoto  extends Remote{
    void inserir(Pessoa p)  throws RemoteException;
    List<Pessoa> listarPessoa() throws RemoteException;
}

```

Classe ServicoRemotoImpl:

```java
package comum;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.List;

public class ServicoRemotoImpl extends UnicastRemoteObject  implements IServicoRemoto{

    private static List<Pessoa> lstPessoa = new ArrayList<>();

    public ServicoRemotoImpl() throws RemoteException {
    }

    private static final long serialVersionUID = 7334161650385718588L;

    @Override
    public void inserir(Pessoa p) throws RemoteException {
        lstPessoa.add(p);
    }
    @Override
    public List<Pessoa> listarPessoa() throws RemoteException {
        return lstPessoa;
    }

}

```

Aula2RMIServidor

Classe Servico

```
1  package servidor;
2
3  import java.rmi.Naming;
4  import java.rmi.RemoteException;
5  import java.rmi.registry.LocateRegistry;
6
7  import comum.ServicoRemotoImpl;
8
9  public class Servico {
10
11     public Servico() throws RemoteException {
12            try {
13               LocateRegistry.createRegistry(8282);
14               System.out.println("INICIO");
15               Naming.rebind("rmi://localhost:8282/pessoa", new ServicoRemotoImpl());
16               System.out.println("FIM");
17            } catch (Exception e) {
18                e.printStackTrace();
19            }
20        }
21        public static void main(String[] args) throws RemoteException {
22           new Servico();
23        }
24 }
25
```

Aula2RMI

Classe AppPrincipal

```
2
3  import java.net.MalformedURLException;
10
11 public class AppPrincipal {
12
13     public static void main(String[] args) {
14
15         try {
16             IServicoRemoto objPessoa = (IServicoRemoto) Naming.lookup("rmi://localhost:8282/pessoa");
17
18             Professor p1 = new Professor();
19
20             p1.setId(1);
21             p1.setNome("Pedro");
22             p1.setTelefone("(12) 1234-5678");
23             p1.setCidade("Araraquara");
24
25             Professor p2 = new Professor();
26
27             p2.setId(2);
28             p2.setNome("Bruno");
29             p2.setTelefone("(11) 4321-7656");
30             p2.setCidade("Araraquara");
31
32             //Inserir Professor
33             objPessoa.inserir(p1);
34             objPessoa.inserir(p2);
35
36             //Listar
37             for (Pessoa professor : objPessoa.listarPessoa()) {
38                 System.out.println("ID: " + professor.getId());
39                 System.out.println("Nome: " + professor.getNome());
40                 System.out.println("Telefone: " + professor.getTelefone());
41                 System.out.println("Cidade: "+ professor.getCidade());
42             }
43
44         } catch (MalformedURLException e) {
45             e.printStackTrace();
46         } catch (RemoteException e) {
47             e.printStackTrace();
48         } catch (NotBoundException e) {
49             e.printStackTrace();
50         }
51     }
52 }
```

Saída:

```
<terminated> AppPrincipal [Java Application] C:\Progr
ID: 1
Nome: Pedro
Telefone: (12) 1234-5678
Cidade: Araraquara
ID: 2
Nome: Bruno
Telefone: (11) 4321-7656
Cidade: Araraquara
```

Atividade:

1) **Criar uma aplicação de Cadastro de pessoa Física (CPF) e Pessoa Jurídica (CNPJ), a Pessoa deve ter: Id, Nome, Telefone, Idade, Endereço, Cidade, Estado, Salário, Nome do Pai e Nome da Mãe.**
2) **Criar as seguintes funcionalidades:**
    a. **Inserir Pessoa**
    b. **Listar Pessoa**
    c. **Gerar Aumento**
        i. **Pessoas com até 20 anos, atribuir um aumento de 10% no salário, entre 20 e 30 anos, atribuir 15%, se for maior que 30 anos, atribuir 20% de aumento.**
    d. **Criar uma rotina para impressão.**
3) **Toda esta estrutura deve ser feita seguindo o exemplo estudado em sala (RMI).**