**public class Server {**

private ServerSocket servsocket;

private int porto;

private Banco b;

public Server (int porto){

this.porto = porto;

this.b = new Banco();}

public void startServer(){

try {this.servsocket = new ServerSocket(this.porto);

while(true){Socket socket = servsocket.accept();

ServerWorker sw = new ServerWorker(socket, b);

new Thread(sw).start();}

} catch …

public static void main(String[] args) {

Server s = new Server(12345);s.startServer();}

**public class Client implements InterfaceBanco{**

private Socket socket;

private ObjectOutputStream out;

private ObjectInputStream in;

public Client(String hostname, int porto){

try { this.socket = new Socket(hostname, porto);

this.out = new ObjectOutputStream(socket.getOutputStream());

this.in = new ObjectInputStream(socket.getInputStream());

} catch …

public int criarConta(double saldo) {

int id = -1;

try{ out.writeObject(OpsBanco.CRIAR\_CONTA);

out.writeDouble(saldo);

out.flush();

id = in.readInt();

} catch …

return id;}}

public static void main(String[] args) {

Client c1 = new Client("127.0.0.1",12345);

c1.criarConta(10);

c1.close();}}

**public class ServerWorker implements Runnable {**

private Socket socket;

private Banco b;

public ServerWorker (Socket socket, Banco bank){

this.socket = socket; this.b = bank;}

public void run() {

try{

ObjectInputStream in = new ObjectInputStream(socket.getInputStream());

ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());

OpsBanco op;

while(true){

double saldo = 0; int id = 0; double valor = 0; op = (OpsBanco) in.readObject();

try{

switch(op){

case CRIAR\_CONTA:

System.out.println(OpsBanco.CRIAR\_CONTA)

saldo = in.readDouble(); id = b.criarConta(saldo);

out.writeInt(id); out.flush(); break;

case FECHAR\_CONTA:

System.out.println(OpsBanco.FECHAR\_CONTA);

id = in.readInt(); saldo = b.fecharConta(id);

out.writeInt(OpsBanco.OK.ordinal()); out.writeDouble(saldo); out.flush(); break;

default:

out.writeInt(OpsBanco.OP\_INVALIDA.ordinal());

out.flush();}

} catch(ContaInvalida ci){

out.writeInt(OpsBanco.CONTA\_INVALIDA.ordinal()); out.flush();}

}

} catch (IOException ioe) {

try{

socket.shutdownOutput(); socket.shutdownInput(); socket.close();}

catch(Exception e){

e.printStackTrace(); }

} catch (ClassNotFoundException ce){

ce.printStackTrace(); }

}

**public class ServerWorker implements Runnable {**

private Socket socket;

private Banco b;

public ServerWorker (Socket socket, Banco bank){

this.socket = socket; this.b = bank;}

public void run() {

try{System.out.println("\nWorker-"+Thread.currentThread().getId());

ObjectInputStream in = new ObjectInputStream(socket.getInputStream());

ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());

OpsBanco op;

while(true){

double saldo = 0; int id = 0; op = (OpsBanco) in.readObject();

try{

switch(op){

case FECHAR\_CONTA:

System.out.println(OpsBanco.FECHAR\_CONTA);

id = in.readInt(); saldo = b.fecharConta(id);

out.writeInt(OpsBanco.OK.ordinal()); out.writeDouble(saldo); out.flush(); break;

default:

System.out.println("\nWorker-"+Thread.currentThread().getId() > op);

out.flush();}

} catch (IOException ioe) {

try{

socket.shutdownOutput(); socket.shutdownInput(); socket.close();}

catch(Exception e){

e.printStackTrace(); }

} catch (ClassNotFoundException ce){

ce.printStackTrace(); }

}

public enum OpsBanco {

FECHAR\_CONTA;}