# 1ªAula Prática

**Objetivo:** Desenvolvimento de aplicações Cliente/Servidor com objetos distribuídos em java RMI e execução da aplicação servidora em máquina virtual na Google Cloud Platform.

**Nota importante:** De acordo com as regras de avaliação definidas no slide 6 do conjunto *CD-01 Apresentação.pdf*, deve submeter, quando disponível na atividade Moodle um pequeno relatório (1 a 2 páginas PDF) descrevendo o trabalho realizado (dificuldades, conclusões, etc.), incluindo um zip com o código desenvolvido. A data limite para entrega é às 23h59 de 08/11/2020

## Exercício 1

Considere um servidor Java RMI que implementa um jogo com as seguintes características:

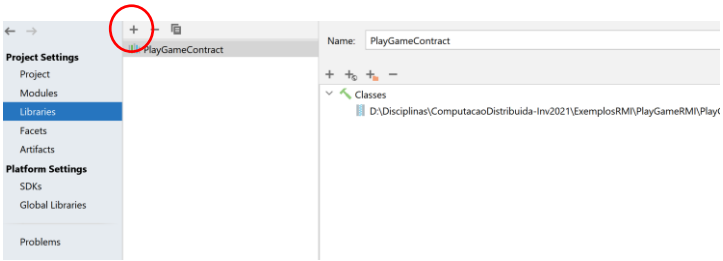
- Existe no servidor a simulação de um rio onde os jogadores podem tentar apanhar coisas incluindo 3 pérolas;
- O rio é simulado como um array bidimensional com 35 posições, onde em cada posição (x, y) existe uma coisa nomeadamente as 3 pérolas;
- O servidor aceita tentativas de qualquer aplicação cliente para tentar encontrar as três pérolas no rio de acordo com o contrato indicado de seguida e disponível no package ***PlayGameContract.jar*** em Java 11, fornecido em anexo;

```
public interface IPlayGame extends Remote {
    Reply playGame(Bet req) throws RemoteException;
}
```

```
public class Bet implements Serializable {
    private int myId; //Id jogador (Nº de aluno)
    public int getMyId() {  }
    public void setMyId(int myId) { }
    private int x;
    private int y;
    public int getX() { }
    public void setX(int x) { }
    public int getY() { }
    public void setY(int y) { }
    public Bet(int myId, int x, int y) {
        setMyId(myId); setX(x); setY(y);
    }
}
```

```
public class Reply implements Serializable {
    private int ntries;
    private boolean success;
    private String thing;
    public int getNtries() {  }
    public void setNtries(int ntries) {  }
    public boolean isSuccess() {  }
    public void setSuccess(boolean success) {  }
    public String getThing() {  }
    public void setThing(String thing) {  }
    public Reply(int ntries,
            boolean success, String thing) {  }
}
```

O servidor já existe e está em execução numa máquina virtual Linux CentoOS 8, da *Google Cloud Platform* com endereço IP 35.230.146.225, que também disponibiliza o serviço de *rmi registry* no porto **7000**. O nome do objeto registado no serviço *rmi registry* para fazer *lookup* é ***GameServer***.
Implemente uma aplicação cliente em Java RMI que permita ao utilizador submeter tentativas para encontrar as coordenadas das 3 pérolas no rio. Note que propositafdamente não é conhecido o comprimento e largura do rio em termos de x e y.
A aplicação cliente deve usar o contrato ***PlayGameContract.jar*** como biblioteca do projeto.



## Exercício 2

Implemente uma aplicação cliente servidor com objetos distribuídos em Java RMI com as seguintes características:

- O Servidor é um agente de leilões e disponibiliza o seguinte contrato:

```
public interface ILeiloes extends Remote {
    // Inicia um leilão de um objeto indicado em objlei e um objeto
    // de callback para receber notificações sobre as licitações efetuadas
    void initLeilao(SomeObject objLei, INotification cb) throws RemoteException;
    // devolve todos os objetos existentes a leilão
    SomeObject[] getAllLeiloes()throws RemoteException;
    // dada o Id de um objeto em leilão permite licitar no leilão associado e passar
    // um objeto de callback para receber notificações sobre as licitações em curso
    void licitar(string Id, float valor, INotification cb) throws RemoteException;
}
// DTO que descreve um objecto em leilão com o valor base para iniciar o leilão
public class SomeObject implements Serializable {
    String Id; // assuma que é unico
    // descrição do objeto
    // valor inicial e atual do leilao
}
// para que a aplicação cliente receba notificações por callback do servidor
public interface INotification extends Remote {
    void sendNotification(String texto) throws RemoteException;
}
```

Tal como no exercício 1, deve existir um desacoplamento entre o cliente e o servidor, devendo estes depender de um contrato existente num JAR. Note que dessa forma os elementos do grupo podem responsabilizar-se por desenvolver cada um a sua aplicação cliente.

*Luís Assunção*