Functionalities implemented

Code in the part1And2 package implements basic part 1&2 requirements.

Code in the extend1_manyCases package implements the first extension: the prosecutor has many cases with associated responses from P2 and the prisoner P1 can sign up to a particular case.

Code in the extend2_courtServices package implements the second extension: after receiving updated sentence, the prisoner can appeal to a court and the courtService will arbitrarily decide on whether to grant this or not by randomly decrease the sentence.

In order to separate solutions and ease the running, each package can be run solely with corresponding function implementation situation.

Design

I use Java RMI as the middleware for this practical.

Remote interfaces and their implementation classes

The interface prosecutor and court extend the Remote class, where abstract methods need to be called remotely are defined, such as testConnection(), mainService (String choice) and appealService (String appealRequest,int originalSentence). These two remote interfaces are implemented by the prosecutorService and courtService class. In this way, these two classes can extend the UnicastRemoteObject class so that their instances can be exported. All the methods (including constructors) will throw RemoteException if the connection fails.

The prosecutorService overrides 3 methods: testConnection() is meant to return a message back to the client, indicating the successful connection between the server and the client. The mainService(String choice) method takes P1's choice, calculates and returns the sentence of the client based on the rule table. In the basic version of program, P2's choice is randomly generated by the server. As for in the extensions, I use a switch solution to deal with 3 different case modes (random, retaliation and tit for tat) the user chooses. At last, the getP2() method is designed to export the P2 choice to be displayed at the client so that the user can know the computer's choice and how the sentence is generated.

Game server

The server defines a single method: registerService(). This method creates stubs of court/prosecutor services and bind them in the local registry on port 8888. Hence, the client can look up this registry and get a reference of corresponding object. All the game processing functions can be done by calling the object's methods. When the client is deployed on other machine, the only effort needed is changing "local host" to the Internet address of the server in the registry looking up statement Naming.lookup("rmi://localhost:8888/pService").

Client

The client has 4 methods (apart from the main method): lookForProsecuterService(), lookForCourtService(), testConnection () and startGame(). In the main method, the first thing is making connection with the server by looking up the registry and getting the services' object reference, calling the testConnection () method of this object. The successful execution of this method indicates that the connection is made. Then the client starts the game: printing instruction of the game, prompting for input and displaying the sentence and appeal results if needed.

Examples

Run the server, and the binding results will be shown as below:

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
>>>>>Remote prosecutorService object is successfully bind to the registry.
>>>>>Remote courtService object is successfully bind to the registry.
```

Knowing the server is standing by, running the client. The game introduction and instruction will be shown as below:

```
Hello from Server.

INSTRUCTION OF PRISONER'S DILEMMA
This is a well-known example of a game representing a social dilemma involving two or more participants.
Each participant represents a prisoner in a cell with no communication with other prisoners.
They are all suspect of a crime and the prosecutor needs to establish the sentence for each of the prisoners.
The prosecutor gives the following choice to each of the prisoners:
   Betray the others by testifying that the others committed the crime
   Cooperate with the others by remaining silent
On the basis of the answers received, the prosecutor decides to discount the sentence according to rules below:
   If the two prisoners cooperate, each player gets 5 years reduction
   If the two prisoners betray each other, each player gets 1 year reduction
   If a prisoner cooperates and the other one betrays, the first gets 2 years reduction and the other one gets 3 years reduction.

Now you are playing this game with a computer.

There are 3 game modes: Random, Retaliation, and Tit for Tat. !!!
Random: This game mode, which is most like a real situation in life,
 makes it so that the computer randomly chooses which path it will choose. It's randomized onto whether or not it will defect or cooperate. This is the most simple game mode as well.
Retaliation: As it sounds,  if you defect, the computer will defect. But if you cooperate, the computer will randomly make choices.
Tit-for-Tat: works like this:
you cooperate, the computer will then cooperate. However, if you defect, the the computer will randomly make choices.

Which game mode would you like to play?
```

Choose a case mode and cooperation/betray, the sentence will be generated and the user can choose if an appeal is needed. Finally, the new sentence will be shown.

The random case mode example is as below:

```
Which game mode would you like to play?

random

You have chosen RANDOM mode. Now type in C for cooperation and B for betray.
c
The computer chose B and you have been sentenced to 2 years.

 If you are not happy with your sentence years, you may appeal to a court to have your discount increased.
 if you want to appeal,type APPEAL; if you don't, type NOAPPEAL and the game is over.
appeal
Your appeal request is now sent to the court.
After the discussion of the court, your sentence is now 1 years. I am afraid you only have ONE appealing chance and the game now is over.
```

The retaliation case mode example is as below:

```
Which game mode would you like to play?
retaliation

You have chosen RETALIATION mode. Now type in C for cooperation and B for betray.
c
The computer chose B and you have been sentenced to 2 years.

 If you are not happy with your sentence years, you may appeal to a court to have your discount increased.
 if you want to appeal,type APPEAL; if you don't, type NOAPPEAL and the game is over.
appeal
Your appeal request is now sent to the court.
After the discussion of the court, your sentence is now 1 years. I am afraid you only have ONE appealing chance and the game now is over.
```

The tit for tat case mode example is as below:

```
Which game mode would you like to play?

tit-for-tat

You have chosen TIT-FOR-TAT mode. Now type in C for cooperation and B for betray.
b
The computer chose B and you have been sentenced to 1 years.

 If you are not happy with your sentence years, you may appeal to a court to have your discount increased.
 if you want to appeal,type APPEAL; if you don't, type NOAPPEAL and the game is over.
noappeal

Process finished with exit code 0
```

As for testing approach, instead of unit tests due to time limits, I just refactored the code by running it with different input.

Evaluation

Since RMI is a Java native, integration of its remote object facilities into a Java application is seamless. RMI-enabled objects can be used as if they live in the local Java environment. RMI is platform-independent (though not language-independent). You can migrate entire objects (i.e. class byte codes are portable) to a remote host via object serialization in RMI. This is not possible in other distributed object architectures such as Common Object Request Broker Architecture (CORB A).

For my system, RMI is sufficient to be a bridge between my server and client. And many clients can look up the registry at the same time and run their clients without extra efforts to deal with multi-threads in the server.

Running instruction

I suggest run the three packages in the order of: part1And2, extend1_manyCases and extend2_courtServices. Take the first one as an example, open a terminal and change the directory to the src, run the server (java part1And2.gameServer). Then open a new terminal (during to the client is on the machine,too), also change the directory to the src, run the client (java part1And2.prisonerClient). Then walk with the instruction. Remember to stop the server(close the terminal) when you move forward to the next extension due to they share the same port.