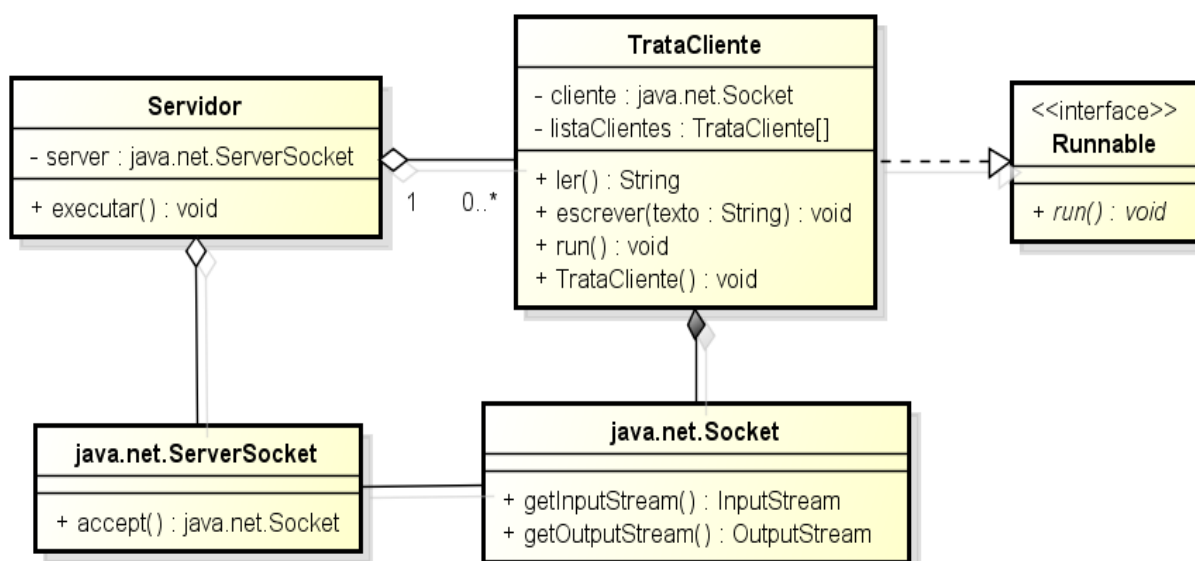


# ATT110 - Programação Orientada a Objetos - Linguagem Java

Home ► My courses ► ATT110-OO-JSDK ► Topic 20 ►  
Exercicio - Servidor Chat para múltiplos clientes ...

## Exercicio - Servidor Chat para múltiplos clientes (Sala de Bate Papo)



powered by Astah

Exercício para criação de servidor chat que permita o atendimento de vários clientes.

Objetivo deste exercício é criar um servidor de chat que permita a conexão de vários clientes, mostrando na console (log) todas as informações que os clientes enviarem pelo canal (Stream).

Para isto deverá seguir as instruções abaixo:

### Na classe TrataCliente

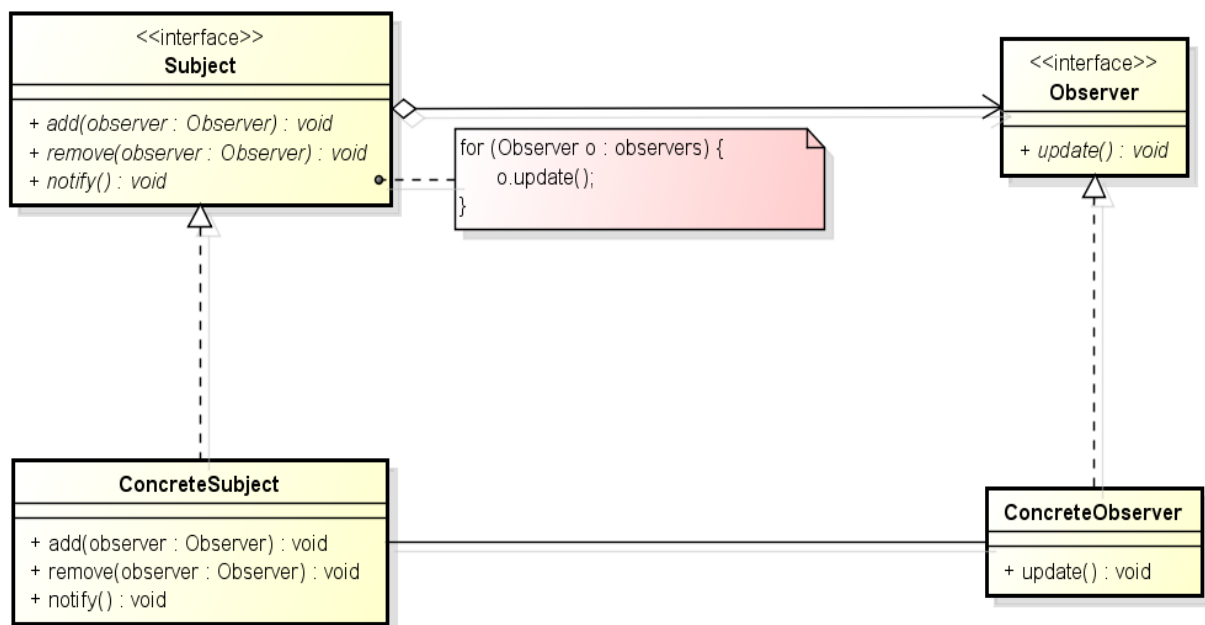
- 1) O construtor da classe `TrataCliente` deve colocar no atributo de instância ( `cliente` ) a referência do objeto `Socket` ( `c` ) recebida como argumento.
- 2) O método `escrever( texto : String )` da classe `TrataCliente` deve pegar o canal `OutputStream()` do objeto `Socket` ( representado pelo atributo `cliente` ) e encaminhar o texto recebido como argumento através dele.
- 3) O método `ler() : String` da classe `TrataCliente` deve pegar o canal `InputStream()` do objeto `Socket` ( representado pelo atributo `cliente` ) e ler os dados deste canal acumulando-os em um texto, até que não existam mais dados a serem lidos, para que seja retornado pelo método.
- 4) O método `run()` deve rodar um loop infinito, que chame o método `ler ( )` imprimindo seu resultado.

### Na classe Servidor

- 5) O construtor deve inicializar o atributo listaCliente
- 6) O método executar() da classe Servidor deve fazer as seguintes tarefas:
  1. Acionar o método accept() do objeto ServerSocket representado pelo atributo de instância chamado server, guardando o retorno em uma variável local do tipo Socket.
  2. Instanciar um novo objeto do tipo TrataCliente passando como parâmetro para o construtor, o objeto do tipo Socket guardado na variável local.
  3. Instanciar um objeto do tipo Thread passando o objeto TrataCliente criado na etapa anterior como parâmetro.
  4. Startar a Thread.
  5. Voltar ao passo a.

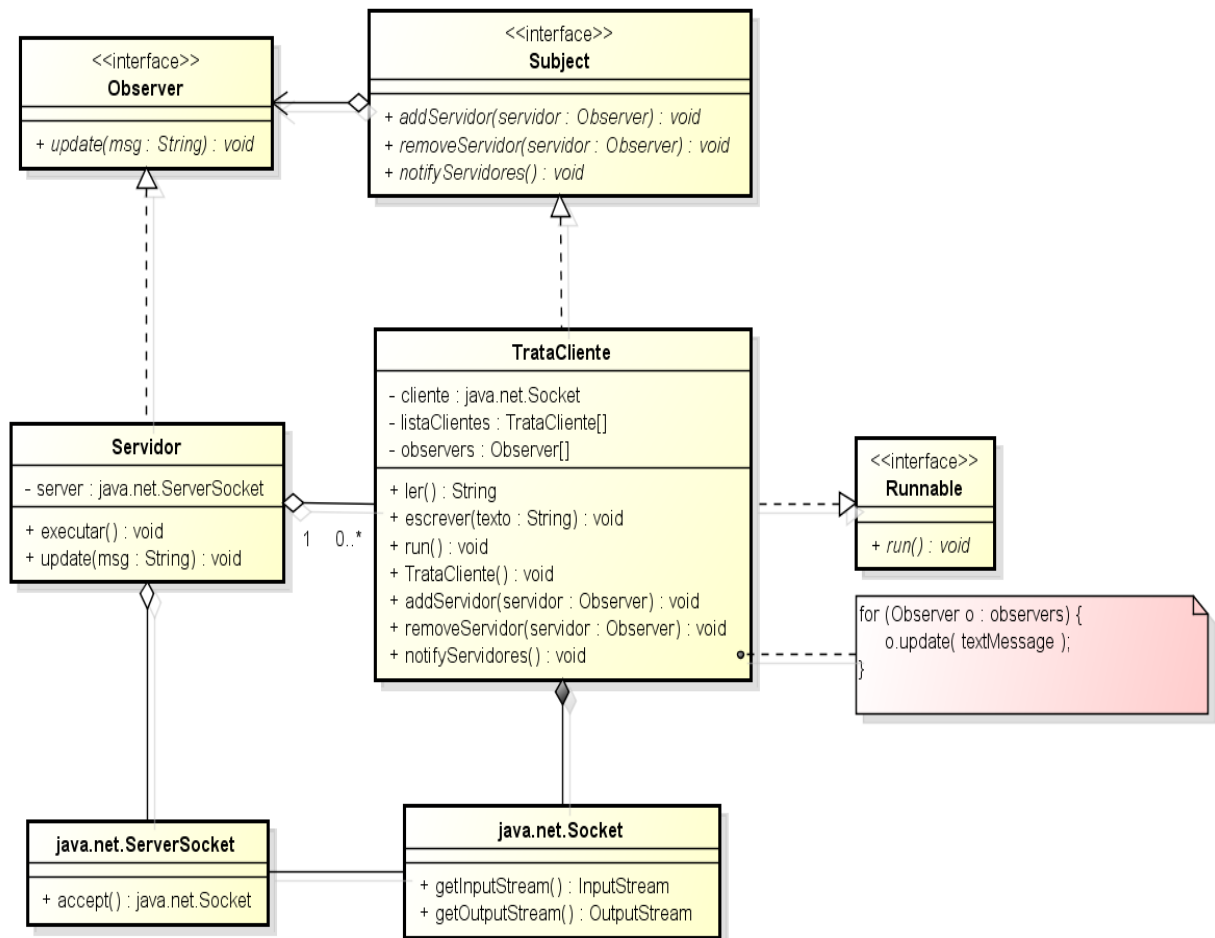
**Aplicação do Padrão Observer para fazer um *broadcast* para todos os clientes quando uma mensagem for recebida**

- 7) Modifique o cenário anterior aplicando o padrão de projetos Observer, para que o objeto Servidor se registre no objeto TrataCliente para saber quando algum cliente mandou alguma mensagem.



powered by Astah

- 8) O diagrama final do sistema ficará da seguinte forma:



powered by Astah

9) Crie as interfaces Observer e Subject conforme o diagrama acima

10) Faça com que a classe TrataCliente implemente a interface Subject

1. Crie um atributo privado para guardar a lista de observers
2. Modifique o construtor para inicializar a lista de observers
3. Modifique o método run() para que invoque o método notifyServidores( <com o texto>) toda vez que receber algum texto do cliente
4. Sobreescreva o método addServidor para que ele adicione o observer recebido como parâmetro na lista de observers
5. Sobreescreva o método addServidor para que ele remova o observer recebido como parâmetro da lista de observers
6. Sobreescreva o método notifyServidores para que ele invoque o método update( <mensagem de Texto>) de todos os observers da lista, conforme abaixo:

```

for (Observer o : observers) {
    o.update(message);
}

```

11) Faça com que a classe Servidor implemente a interface Observer

1. Modifique o método executar para adicionar este objeto servidor no objeto TrataCliente através da execução do método <objeto TrataCliente>.addServidor( this )
2. Sobreescreva o método update( String : texto ) para invocar o método escrever de todos os objetos do tipo TrataCliente que existem na listaClientes conforme abaixo:

```

for (TrataCliente temp : listaClientes) {
    temp.escrever( texto );
}


```

# Grading summary

Participants	76
Submitted	0
Needs grading	0

[View all submissions](#)

[Grade](#)

 Moodle Docs for this page

You are logged in as Admin User (Log out)  
ATT110-OO-JSDK