

Parte 1: TCP

Sumário:

comunicação entre 2 servidores utilizando o serviço TCP. Foi utilizada a linguagem Kotlin e a biblioteca Socket. A porta que o servidor escuta é “6789” enquanto o IP do cliente é o mesmo da máquina. “127.0.0.1”

Observação: Kotlin gera o mesmo byte code que Java, sendo portanto exatamente a mesma biblioteca, sem um único método diferente.

Descrição:

Foram implementadas 4 classes: Client, Server, ValidationManager e GameAction.

- Client e Server são classes que lidam gerar dados para começar o jogo e, no nosso contexto, estabelecer conexões entre si
- ValidationManager é uma classe que lida com erros de input que o usuário pode vir a cometer ao escrever algo não condizente á uma solicitação no terminal
- GameAction é a classe que une todas as ações que o jogo possui. Atacar, esperar ataque, gerar feedback ao oponente. Seus métodos são compartilhados por Client e Server, poupando linhas de código

Não obstante algumas abstrações foram criadas para aumentar a legibilidade do código. São eles: Ships[Enum], Orientations[Enum], FeedbackTypes[Enum] e Point[Data_class]

O trabalho dos Enums são representar situações do jogo, que usualmente são representadas por Inteiros(forá do mundo Kotlin/Swift) como palavras que, por trás, tem um valor inteiro e que podem ser recebidos como parâmetros em métodos.

Representação:

Ships -> Barcos

Orientations -> Left,Right,Up,Down

FeedbackTypes -> Feedbacks direcionados ao cliente

Point -> Ponto (x,y)

Os métodos foram pensados para serem o mais intuitivos possíveis. Portanto vou explicar os principais, que acredito serem mais influentes no jogo:

setupShip -> Criar navios

drawShip -> Colocar navio no Oceano

doAttack -> Atacar oponente

waitAttack -> Esperar por ataque

receiveAttack -> Receber/validar ataque

waitFeedback -> Esperar resposta de ataque

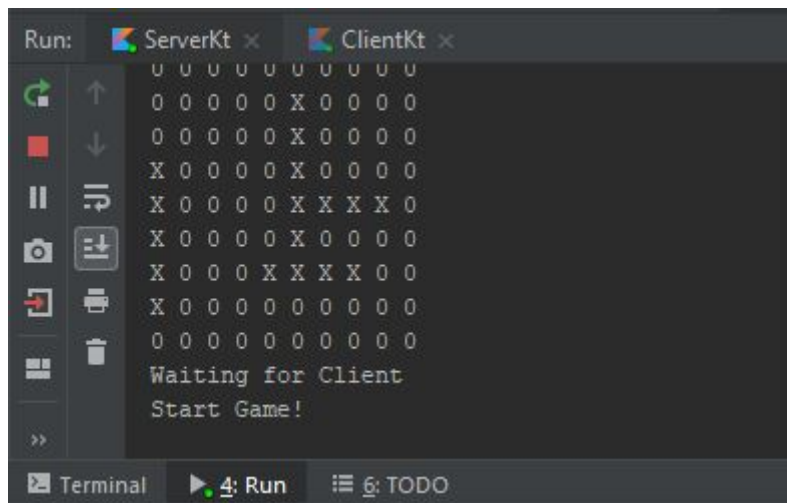
sendFeedback -> Enviar resposta do ataque

Decisões de implementação:

Separação entre classes distintas, criação de uma classe para lidar com erros de usuários

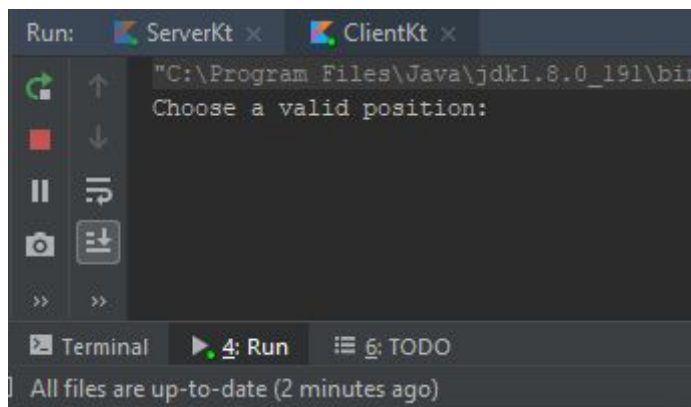
Testes:

Início do jogo(Server)



```
Run: ServerKt x ClientKt x
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 X 0 0 0 0
0 0 0 0 0 X 0 0 0 0
X 0 0 0 0 X 0 0 0 0
X 0 0 0 0 X X X X 0
X 0 0 0 0 X 0 0 0 0
X 0 0 0 X X X X 0 0
X 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
Waiting for Client
Start Game!
```

Início(Client)

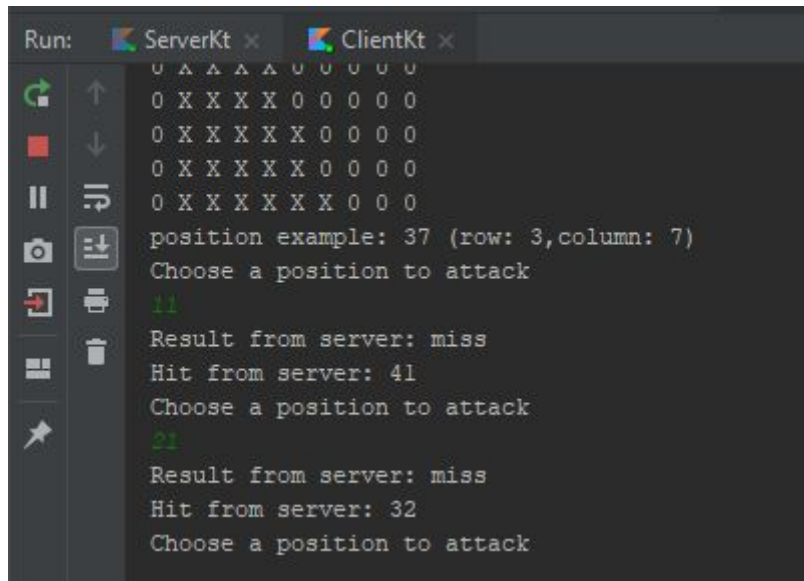


```
Run: ServerKt x ClientKt x
"C:\Program Files\Java\jdk1.8.0_191\bin
Choose a valid position:
```

Barcos criados(Client)

```
ServerKt x ClientKt x
0 X X X X X 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
Choose a valid position:
81
Choose a valid orientation:
1-NORTH
2-SOUTH
3-WEAST
4-EAST
3
0 0 0 0 0 0 0 0 0 0
0 X X X X 0 0 0 0 0 0
0 X X X 0 0 0 0 0 0
0 X X X 0 0 0 0 0 0
0 X X X 0 0 0 0 0 0
0 X X X X 0 0 0 0 0
0 X X X X 0 0 0 0 0
0 X X X X 0 0 0 0 0
0 X X X X 0 0 0 0 0
0 X X X X X 0 0 0 0
0 X X X X X 0 0 0 0
0 0 0 0 0 0 0 0 0 0
Choose a valid position:
81
Choose a valid orientation:
1-NORTH
2-SOUTH
3-WEAST
4-EAST
3
0 0 0 0 0 0 0 0 0 0
0 X X X 0 0 0 0 0 0
0 X X X 0 0 0 0 0 0
0 X X X 0 0 0 0 0 0
0 X X X X 0 0 0 0 0
0 X X X X 0 0 0 0 0
0 X X X X 0 0 0 0 0
0 X X X X X 0 0 0 0
0 X X X X X 0 0 0 0
0 X X X X X 0 0 0 0
0 X X X X X X 0 0 0
position example: 37 (row: 3,column: 7)
Choose a position to attack
```

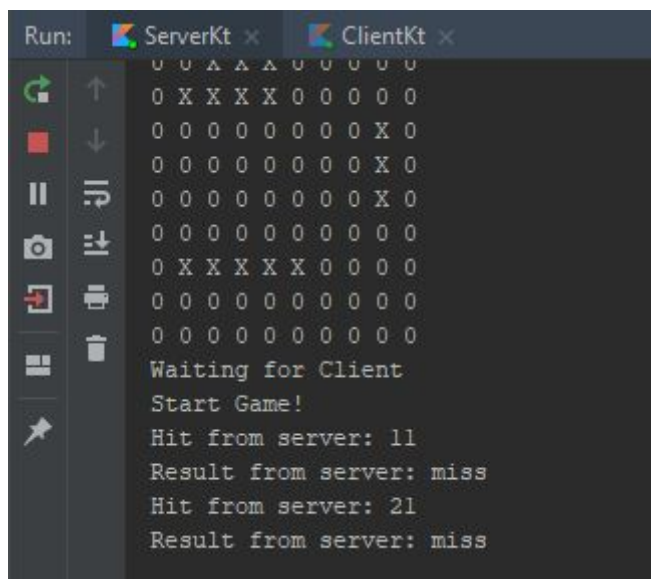
Executando ataques e recebendo feedbacks(Client)



The screenshot shows the IntelliJ Run console with two tabs: 'ServerKt' and 'ClientKt'. The 'ClientKt' tab is active, displaying the following text:

```
0 0 0 0 0 0 0 0 0 0
0 X X X X 0 0 0 0 0
0 X X X X X 0 0 0 0
0 X X X X X 0 0 0 0
0 X X X X X X 0 0 0
position example: 37 (row: 3,column: 7)
Choose a position to attack
11
Result from server: miss
Hit from server: 41
Choose a position to attack
21
Result from server: miss
Hit from server: 32
Choose a position to attack
```

Executando ataques(aleatórios) e recebendo feedbacks(Server)



The screenshot shows the IntelliJ Run console with two tabs: 'ServerKt' and 'ClientKt'. The 'ServerKt' tab is active, displaying the following text:

```
0 0 0 0 0 0 0 0 0 0
0 X X X X 0 0 0 0 0
0 0 0 0 0 0 0 0 X 0
0 0 0 0 0 0 0 0 X 0
0 0 0 0 0 0 0 0 X 0
0 0 0 0 0 0 0 0 0 0
0 X X X X X 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
Waiting for Client
Start Game!
Hit from server: 11
Result from server: miss
Hit from server: 21
Result from server: miss
```

Conclusão: A comunicação entre client/servidor foi um sucesso. O processo como um todo foi um ótimo aprendizado para entender como funciona um pouco da biblioteca Socket e como, por trás, é feita uma comunicação TCP. IDE utilizada: IntelliJ.

