

Exercícios Aula 23

1. No tópico **Cliente-Servidor Java Sockets Bate-Papo**, criar um projeto denominado **ProjetoBatepapo**, na IDE de sua preferência, digitar as classes fornecidas **SocketClient.java**, **ServidorBatepapo.java** e **ClienteBatepapo.java**, distribuídas entre as páginas 11 à 21 deste material, executar, analisar, concluir e registrar o funcionamento das mesmas;

Executando o código, primeiro iniciando o servidor e depois os clientes, qualquer mensagem enviada no cliente é exibida no servidor. Caso haja mais de um cliente, as mensagens enviadas em um cliente são exibidas em todos os clientes.

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
			<pre>Copyright (C) Microsoft Corporation. T Experimente a nova plataforma cruzada PS C:\Users\21.01837-5\Downloads\Aula_ eDetailsInExceptionMessages' '-cp' 'C: 1f8732c0e8cb940952abd03176bbd\redhat.j *v*v*v* CONSOLE DO SERVIDOR *v*v*v* Servidor iniciado na porta: 4000 Aguardando conexão de um cliente! Conectado com /127.0.0.1:53758! <- Cliente /127.0.0.1:53758: oi</pre>	<pre>Windows PowerShell Copyright (C) Microsoft Corporation. T Experimente a nova plataforma cruzada PS C:\Users\21.01837-5\Downloads\Aula_ deDetailsInExceptionMessages' '-cp' 'C c21f8732c0e8cb940952abd03176bbd\redhat *^*^*^* CONSOLE DO CLIENTE *^*^*^* Conectado com /127.0.0.1:4000! Digite uma mensagem (ou <sair> para fi <- oi</pre>

2. O que acontece se na aplicação do Exercício 1, o número de **Clientes** instanciados for elevado? Há um limite para a quantidade de **Clientes** na aplicação do **Servidor** fornecida? Justifique todas as respostas!

Devido ao uso de threads, um número elevado de clientes instanciados não irá comprometer o funcionamento da aplicação, e todos os clientes verão e enviarão mensagens a todos conectados.

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
imeiro -< Cliente /127.0.0.1:53778: se gundo Conectado com /127.0.0.1:53838! -< Cliente /127.0.0.1:53838: qu arto Conectado com /127.0.0.1:54407! -< Cliente /127.0.0.1:54407: qu into	para finalizar): -< segundo -> Cliente /127.0.0.1:53838: qu arto Digite uma mensagem (ou <ctrl> para finalizar): -< -> Cliente /127.0.0.1:54407: qu into Digite uma mensagem (ou <ctrl> para finalizar): Ctrl	<-segundo -> Cliente /127.0.0.1:53838: qu arto Digite uma mensagem (ou <ctrl> para finalizar): -< -> Cliente /127.0.0.1:54407: qu into Digite uma mensagem (ou <ctrl> para finalizar): Ctrl	Digite uma mensagem (ou <ctrl> para finalizar): -> Cliente /127.0.0.1:53838: qu arto Digite uma mensagem (ou <ctrl> para finalizar): -< -> Cliente /127.0.0.1:54407: qu into Digite uma mensagem (ou <ctrl> para finalizar): Ctrl	***** CONSOLE DO CLIENTE ***** *** Conectado com /127.0.0.1:4000! Digite uma mensagem (ou <ctrl> para finalizar): -< quarto -> Cliente /127.0.0.1:54407: qu into Digite uma mensagem (ou <ctrl> para finalizar): -< quinto ***** CONSOLE DO CLIENTE ***** **** Conectado com /127.0.0.1:4000! Digite uma mensagem (ou <ctrl> para finalizar): -< quinto

O limite de clientes na aplicação é definido por quantas threads o hardware do servidor aguenta.

3. Com base no **ProjetoBatepapo** do Exercício 1 deste material, executar, somente, a classe **ClienteBatePapo.java**, verificar, registrar e explicar, em detalhes, o ocorrido;

Uma exceção ocorre e a execução do programa acaba, pois não há nenhum servidor em execução para o cliente se conectar.

```
PS C:\Users> c:; cd 'c:\Users\21.01837-5\Downloads\Aula_23'; & 'S:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '732c0e8cb940952abd03176bbd\redhat.java\jdt_ws\Aula_23_5ca8b797\bin' 'Aula_23.ClienteBatepapo'
***** CONSOLE DO CLIENTE *****
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "Aula_23.SocketCliente.close()" because "this.clientSocket" is null
    at Aula_23.ClienteBatepapo.start(ClienteBatepapo.java:22)
    at Aula_23.ClienteBatepapo.main(ClienteBatepapo.java:49)
PS C:\Users\21.01837-5\Downloads\Aula_23>
```

4. Com base no **ProjetoBatepapo** do Exercício 1 deste material, executar a classe **ServidorBatepapo.java**, verificar, registrar e explicar, em detalhes, a operação do **Servidor**, sem sair dele;

O servidor entrará em execução e ficará esperando a conexão de um cliente até que haja uma conexão.

```
PS C:\Users\21.01837-5\Downloads\Aula_23> & 'S:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\21.01837-5\AppData\Roaming\Code\User\workspaceStorage\ec21a_23_5ca8b797\bin' 'Aula_23.ServidorBatepapo'
***** CONSOLE DO SERVIDOR *****
Servidor iniciado na porta: 4000
Aguardando conexão de um cliente!
```

5. Com base no **ProjetoBatepapo** do Exercício 1 deste material, com o **Servidor** em operação, executar uma primeira instância da classe **ClienteBatepapo.java**, digitando algumas mensagens no seu console e, por fim, sair do **Cliente**, sem sair do **Servidor**. Verificar, registrar e explicar, em detalhes, as operações do **Cliente** e do **Servidor**;

O cliente se conectará com o servidor e enviará mensagens, às quais o servidor vai exibir sem problemas. Quando o cliente se desconectar, o servidor continuará em execução esperando um próximo cliente se conectar.

```
PS C:\Users\21.01837-5\Downloads\Aula_23> & 'S:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\21.01837-5\AppData\Roaming\Code\User\workspaceStorage\ec21a_23_5ca8b797\bin' 'Aula_23.ServidorBatepapo'
***** CONSOLE DO SERVIDOR *****
Servidor iniciado na porta: 4000
Aguardando conexão de um cliente!
Conectado com /127.0.0.1:54770!
<- Cliente /127.0.0.1:54770: Mensagem 1
<- Cliente /127.0.0.1:54770: Mensagem 2
<- Cliente /127.0.0.1:54770: Mensagem Final.

PS C:\Users\21.01837-5\Downloads\Aula_23> & 'S:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\21.01837-5\AppData\Roaming\Code\User\workspaceStorage\ec21a_23_5ca8b797\bin' 'Aula_23.ClienteBatepapo'
***** CONSOLE DO CLIENTE *****
Conectado com /127.0.0.1:4000!
Digite uma mensagem (ou <sair> para finalizar):
<- Mensagem 1
<- Mensagem 2
<- Mensagem Final.
<- sair
Cliente finalizado!
PS C:\Users\21.01837-5\Downloads\Aula_23>
```

6. Com base no **ProjetoBatepapo** do Exercício 1 deste material, com o **Servidor** em operação, executar, novamente, uma primeira instância da classe **ClienteBatepapo.java**, digitando algumas mensagens no seu console e, por fim, não sair do **Cliente**, nem do **Servidor**. Verificar, registrar e explicar, em detalhes, as operações do **Cliente** e do **Servidor**;

A conexão acontece, as mensagens do cliente são exibidas no servidor, e ambos processos continuam em execução, pois nenhum dos dois recebeu ordem de finalizar execução.

```
PS C:\Users\21.01837-5\Downloads\Aula_23> c:; cd 'c:\Users\21.01837-5\Downloads\Aula_23'; & 'S:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\21.01837-5\AppData\Roaming\Code\User\workspaceStorage\ec21f8732c0e8cb940952abd03176bbd\redhat.java\jdt_ws\Aula_23_5ca8b797\bin' 'Aula_23.ServidorBatepapo'
*v*v*v* CONSOLE DO SERVIDOR *v*v*v*
Servidor iniciado na porta: 4000
Aguardando conexão de um cliente!
Conectado com /127.0.0.1:54858!
<- Cliente /127.0.0.1:54858: Mensagem 1
<- Cliente /127.0.0.1:54858: Mensagem 2
<- Cliente /127.0.0.1:54858: Mensagem 3
[]

PS C:\Users\21.01837-5\Downloads\Aula_23> c:; cd 'c:\Users\21.01837-5\AppData\Roaming\Code\User\workspaceStorage\ec21f8732c0e8cb940952abd03176bbd\redhat.java\jdt_ws\Aula_23_5ca8b797\bin' 'Aula_23.ClienteBatepapo'
***** CONSOLE DO CLIENTE *****
Conectado com /127.0.0.1:4000!
Digite uma mensagem (ou <sair> para finalizar):
<- Mensagem 1
<- Mensagem 2
<- Mensagem 3
<- []
```

7. Com base no **ProjetoBatepapo** do Exercício 1 deste material, com o **Servidor** e a primeira instância do **Cliente** em operação, executar uma segunda instância da classe **ClienteBatepapo.java**, digitando algumas mensagens no seu console e no console da instância do outro **Cliente**, alternadamente e, por fim, não sair dos **Clientes**, nem do **Servidor**. Verificar, registrar e explicar, em detalhes, as operações dos **Clientes** e do **Servidor**;

Os dois clientes se conectam com o servidor, com as mensagens de ambos sendo exibidas no servidor e cada cliente também exibindo as mensagens recebidas do outro (primeiro exhibe as mensagens do segundo cliente e vice versa)

```
*v*v*v* CONSOLE DO SERVIDOR *v*v*v*
Servidor iniciado na porta: 4000
Aguardando conexão de um cliente!
Conectado com /127.0.0.1:54858!
<- Cliente /127.0.0.1:54858: Mensagem 1
<- Cliente /127.0.0.1:54858: Mensagem 2
<- Cliente /127.0.0.1:54858: Mensagem 3
Conectado com /127.0.0.1:54905!
<- Cliente /127.0.0.1:54905: Mensagem 1, cliente 2
<- Cliente /127.0.0.1:54858: Olá cliente 2!
<- Cliente /127.0.0.1:54905: Olá cliente 1
[]

Digite uma mensagem (ou <sair> para finalizar):
<- Mensagem 1
<- Mensagem 2
<- Mensagem 3
<-
-> Cliente /127.0.0.1:54905: Mensagem 1, cliente 2
Digite uma mensagem (ou <sair> para finalizar):
<-Olá cliente 2!
<-
-> Cliente /127.0.0.1:54905: Olá cliente 1
Digite uma mensagem (ou <sair> para finalizar):
<-[]

'-cp' 'C:\Users\21.01837-5\AppData\Roaming\Code\User\workspaceStorage\ec21f8732c0e8cb940952abd03176bbd\redhat.java_5ca8b797\bin' 'Aula_23.ClienteBatepapo'
***** CONSOLE DO CLIENTE *****
Conectado com /127.0.0.1:4000!
Digite uma mensagem (ou <sair> para finalizar):
<- Mensagem 1, cliente 2
<-
-> Cliente /127.0.0.1:54858: Olá cliente 2!
Digite uma mensagem (ou <sair> para finalizar):
<-Olá cliente 1
<- []
```

8. Com base no **ProjetoBatepapo** do Exercício 1 deste material, com o **Servidor** e as duas instâncias dos **Clientes** em operação, executar uma terceira instância da classe **ClienteBatepapo.java**, digitando algumas mensagens no seu console e nos consoles das instâncias dos outros **Clientes**, alternadamente e, por fim, não sair dos **Clientes**, nem do **Servidor**. Verificar, registrar e explicar, em detalhes, as operações dos **Clientes** e do **Servidor**;

O terceiro cliente consegue se conectar com sucesso ao servidor e o servidor exibe as mensagens dos clientes 1, 2 e 3. Os três clientes são capazes de exibir as mensagens de todos os outros clientes, então há comunicação entre os três simultaneamente.

<pre> Conectado com /127.0.0.1:54985! -> Cliente /127.0.0.1:54985: Mensagem 1, cliente 2 - Cliente /127.0.0.1:54858: Oi cliente 2! - Cliente /127.0.0.1:54985: Oi cliente 1 Conectado com /127.0.0.1:54957! -> Cliente /127.0.0.1:54957: Mensagem 1, cliente 3 - Cliente /127.0.0.1:54858: Oi cliente 3 - Cliente /127.0.0.1:54985: Oi cliente 3! - Cliente /127.0.0.1:54957: Oi clientes 1 e 2 </pre>	<pre> -<- -> Cliente /127.0.0.1:54957: Mensagem 1, cliente 3 - Digite uma mensagem (ou <saír> para finalizar): -<- -> Cliente /127.0.0.1:54985: Oi cliente 3! - Digite uma mensagem (ou <saír> para finalizar): -<- -> Cliente /127.0.0.1:54957: Oi clientes 1 e 2 - Digite uma mensagem (ou <saír> para finalizar): -<- </pre>	<pre> -<- -> Cliente /127.0.0.1:54957: Mensagem 1, cliente 3 - Digite uma mensagem (ou <saír> para finalizar): -<- -> Cliente /127.0.0.1:54858: Oi cliente 3 - Digite uma mensagem (ou <saír> para finalizar): -<- -> Cliente /127.0.0.1:54957: Oi clientes 1 e 2 - Digite uma mensagem (ou <saír> para finalizar): -<- </pre>	<pre> Conectado com /127.0.0.1:4800! - Digite uma mensagem (ou <saír> para finalizar): -<- - Mensagem 1, cliente 3 -<- - Cliente /127.0.0.1:54858: Oi cliente 3 - Digite uma mensagem (ou <saír> para finalizar): -<- - Cliente /127.0.0.1:54985: Oi cliente 3! - Digite uma mensagem (ou <saír> para finalizar): -<- - Clientes 1 e 2 </pre>
--	--	---	---

9. Com base no **ProjetoBatepapo** do Exercício 1 deste material, com o **Servidor** e as três instâncias dos **Clientes** em operação, encerrar a execução das instâncias dos **Clientes**, uma por vez, digitando **<sair>** no console para cada uma delas, na ordem **Cliente 1**, **2** e **3** das instâncias. Verificar, registrar e explicar, em detalhes, as operações dos **Clientes** e do **Servidor**;

A saída de cada cliente individual não compromete o funcionamento do servidor nem o dos outros clientes, pois os clientes 2 e 3 após a saída do primeiro ainda se comunicam, e o servidor exibe as mensagens de todos. Quando os três são finalizados, o servidor continua em execução, aguardando clientes.

<pre> - Cliente /127.0.0.1:54858: Oi cliente 2! - Cliente /127.0.0.1:54905: Oi cliente 1 Conectado com /127.0.0.1:54957! - Cliente /127.0.0.1:54957: Mensagem 1, cliente 3 - Cliente /127.0.0.1:54858: Oi cliente 3 - Cliente /127.0.0.1:54905: Oi cliente 3! - Cliente /127.0.0.1:54957: Oi clientes 1 e 2 - Cliente /127.0.0.1:54858: tchau - Cliente /127.0.0.1:54905: tchau - Cliente /127.0.0.1:54957: tchau </pre>	<pre> Digite uma mensagem (ou <saír> para finalizar): <-Oi cliente 3 <- -> Cliente /127.0.0.1:54905: Oi cliente 3! Digite uma mensagem (ou <saír> para finalizar): <- -> Cliente /127.0.0.1:54957: Oi clientes 1 e 2 Digite uma mensagem (ou <saír> para finalizar): <-tchau <- saír Cliente finalizado! PS C:\Users\21.018337-5\Downloads\Aula 23> </pre>	<pre> Digite uma mensagem (ou <saír> para finalizar): <-Oi cliente 3! <- -> Cliente /127.0.0.1:54957: Oi clientes 1 e 2 Digite uma mensagem (ou <saír> para finalizar): <- -> Cliente /127.0.0.1:54858: tchau Digite uma mensagem (ou <saír> para finalizar): <-tchau <- saír Cliente finalizado! PS C:\Users\21.018337-5\Downloads\Aula 23> </pre>	<pre> Digite uma mensagem (ou <saír> para finalizar): <-Oi clientes 1 e 2 <- -> Cliente /127.0.0.1:54858: tchau Digite uma mensagem (ou <saír> para finalizar): <- -> Cliente /127.0.0.1:54905: tchau Digite uma mensagem (ou <saír> para finalizar): <-tchau <- saír Cliente finalizado! PS C:\Users\21.018337-5\Downloads\Aula 23> </pre>
--	--	---	---

10. Com base no **ProjetoBatepapo** do Exercício 1 deste material, ainda com a primeira instância do **Servidor** em operação, executar, a seguir, uma segunda instância da classe **Server.java**. Verificar, registrar e explicar, em detalhes, as operações dos **Servidores**;

A segunda instância do servidor não pode ser executada, pois a porta configurada já está em uso pelo primeiro servidor e é impossível usar a mesma porta em duas execuções diferentes.

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
<p>Copyright (C) Microsoft Corporation. Todos os direitos reservados.</p> <p>Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6</p> <pre>PS C:\Users\Z1-01837-5\Downloads\Aula_23> & "C:\Program Files\Eclipse Adoptium\jdk-21.0.2.13-hotspot\bin\java.exe" -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\Z1-01837-5\AppData\Local\Temp\AUser\WorkSpace\storage\3669df5912b5cf0232d26b6584da4f6\vedhat.java\jdk_vs_Aula_23_Scabb797\bin\" "Aul a_23_ServidorBatepaço" ^ ^*^*^*^* CONSOLE DO SERVIDOR ^*^*^*^* Servidor iniciado na porta: 4000 Aguardando conexão de um cliente! ^ ^</pre>				
<p>Copyright (C) Microsoft Corporation. Todos os direitos reservados.</p> <p>Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6</p> <pre>PS C:\Users\Z1-01837-5\Downloads\Aula_23> & "C:\Program Files\Eclipse Adoptium\jdk-21.0.2.13-hotspot\bin\java.exe" -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\Z1-01837-5\AppData\Local\Temp\AUser\WorkSpace\storage\3669df5912b5cf0232d26b6584da4f6\vedhat.java\jdk_vs_Aula_23_Scabb797\bin\" " Aula_23_ServidorBatepaço" ^ ^*^*^*^* CONSOLE DO SERVIDOR ^*^*^*^* Erro ao iniciar servidor: Address already in use: bind Servidor finalizado PS C:\Users\Z1-01837-5\Downloads\Aula_23></pre>				