09

Mãos à obra: BlockingQueue no Servidor

Uma vez que conhecemos as principais funcionalidades da `BlockingQueue`, podemos voltar ao nosso projeto **servidor-tarefas** para implementar o **produtor** e o **consumidor** da fila.

1) O primeiro passo é criar a `ArrayBlockingQueue` dentro da classe `ServidorTarefa`. Inicialize o `ArrayBlockingQueue` dentro do construtor com uma capacidade pequena de 2 elementos:

```
public class ServidorTarefas {

    private ServerSocket servidor;
    private ExecutorService threadPool;
    private AtomicBoolean estaRodando;
    private BlockingQueue<String> filaComandos; //novo atributos

    public ServidorTarefas() throws IOException {

        System.out.println("---- Iniciando Servidor ----");
        this.servidor = new ServerSocket(12345);
        this.threadPool = Executors.newFixedThreadPool(4, new FabricaDeThreads());
        this.estaRodando = new AtomicBoolean(true);
        this.filaComandos = new ArrayBlockingQueue<>(2); //crianda a fila com capacidade de 2 e
    }

    // restante do código
}
```

2) Uma vez criada a fila, podemos passá-la para a classe `DistribuirTarefas`. Adicione mais um parâmetro no construtor e modifique-o na classe:

```
// método rodar() da classe ServidorTarefas, passaremos a fila como segundo parâmetro
DistribuirTarefas distribuidor = new DistribuirTarefas(threadPool, filaComandos, socket, this);
```

```
public class DistribuirTarefas implements Runnable {

    //outros atributos omitidos
    private BlockingQueue<String> filaComandos; //novo

    public DistribuirTarefas(ExecutorService threadPool, BlockingQueue<String> filaComandos, So
        this.threadPool = threadPool;
        this.filaComandos = filaComandos; //novo
        this.socket = socket;
        this.servidor = servidor;
    }

    // restante do código
}
```

3) No `switch` da classe `DistribuirTarefas`, crie um novo `case` que *oferece* o comando à fila:

```
case "c3" : {
    this.filaComandos.put(comando); //lembrando, bloqueia se tiver cheia
    saidaCliente.println("Comando c3 adicionado na fila");
    break;
}
```

4) Vamos consumir os comandos, ou seja criar a thread que "fica de olho na fila" e tira os elementos dela! Para tal, crie uma nova tarefa utilizando a interface `Runnable` mesmo, pois não estamos interessados em devolver algo. A tarefa deve receber a nossa fila de comandos:

```
public class TarefaConsumir implements Runnable {

    private BlockingQueue<String> filaComandos;

    public TarefaConsumir(BlockingQueue<String> filaComandos) {
        this.filaComandos = filaComandos;
    }

    @Override
    public void run(){
        //aqui vem mais
    }
}
```

5) No método `run()` da classe `TarefaConsumir`, use o método `take()` para verificar se existe um novo comando:

```
@Override
public void run() {

        try {
            String comando = null;

            //enquanto existe um novo comando, lembrando take() bloqueia
            while ((comando = filaComandos.take()) != null) {
                System.out.println("Consumindo comando " + comando + ", " + Thread.currentThrea
                Thread.sleep(20000); //demorando 20s para consumir
            }
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
}
```

6) Com a tarefa criada, vamos inicializar algumas threads com essa tarefa para consumir os comandos :) Na classe `ServidorTarefas` adicione um método auxiliar:

```
// na classe ServidorTarefas
private void iniciarConsumidores() {
    int qtdCondumidores = 2;
    for (int i = 0; i < qtdCondumidores; i++) {
```

```
            TarefaConsumir tarefa = new TarefaConsumir(filaComandos);
            this.threadPool.execute(tarefa);
        }
    }
```

Esse método inicializa e executa dois consumidores pelo pool.

7) Chamar o método `iniciarConsumidores` no construtor da classe `ServidorTarefas` :

```
// no construtor da classe ServidorTarefas
public ServidorTarefas() throws IOException {
    System.out.println("---- Iniciando Servidor ----");
    this.servidor = new ServerSocket(12345);
    this.threadPool = Executors.newFixedThreadPool(4, new FabricaDeThreads());
    this.estaRodando = new AtomicBoolean(true);
    this.filaComandos = new ArrayBlockingQueue<>(2);
    iniciarConsumidores(); //novo
}
```

8) Inicialize o servidor e rode o cliente (parando todos os consoles antes). Envie pelo cliente o comando `c3` pelo menos duas vezes. Os consumidores devem *pegar e consumir os comandos* da fila. Fique atento ao console: