04

Limpando banheiro

Transcrição

Acordar os threads que estão no estado `waiting` chama-se **notificar**. Essa notificação deve ser feita através de um novo thread que executará a tarefa de limpeza. Após ter terminado a limpeza, é preciso avisar os convidados!

Na classe `Banheiro`, implementaremos um novo método, chamado `limpa()`, que muda o booleano que criamos anteriormente. Além disso, vamos notificar todos os threads através do método `notifyAll()`:

```
public void limpa() {
    this.ehSujo = false;
    this.notifyAll();
}
```

Falta ainda criar a tarefa de limpeza e inicializá-la na classe `Principal`. Segue o código:

```
public class TarefaLimpeza implements Runnable {

    private Banheiro banheiro;

    public TarefaLimpeza(Banheiro banheiro) {
        this.banheiro = banheiro;
    }

    public void run() {
        this.banheiro.limpa();
    }
}
```

E no método `main` da classe `Principal`:

```
//outros threads omitidos
Thread limpeza = new Thread(new TarefaLimpeza(banheiro), "Limpeza");

//inicialização dos outros threads omitida
limpeza.start();
```

Vamos testar o nosso código, executando a classe `Principal`. Para nossa surpresa, recebemos uma exceção `IllegalMonitorStateException`!

Analisando esses dados, podemos ver que recebemos uma exceção porque chamamos o método `notifyAll()`. Qual é o problema? Pensando na vida real, faz sentido alguém limpando o banheiro quando ele está ocupado? Claro que não! E no mundo Java isso não é diferente... Só podemos limpar o banheiro e notificar os outros threads quando estamos com a chave em mãos. Ou seja, só podemos chamar `notifyAll()` dentro de um bloco sincronizado.

Vamos modificar o nosso código e também imprimir algumas informações a mais para acompanhar mais fácil a "limpeza":

```
public void limpa() {

    String nome = Thread.currentThread().getName();

    System.out.println(nome + " batendo na porta");

    synchronized (this) {

        System.out.println(nome + " entrando no banheiro");

        if (!this.ehSujo) {
            System.out.println(nome + ", não está sujo, vou sair");
            return;
        }

        System.out.println(nome + " limpando o banheiro");
        this.ehSujo = false;

        try {
            Thread.sleep(13000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        this.notifyAll();

        System.out.println(nome + " saindo do banheiro");
    }
}
```

Agora já podemos testar código e todos os convidados deveriam ter acesso ao banheiro. Segue a saída do console, para simplificar com apenas dois convidados:

```
Limpeza batendo na porta
João batendo na porta
Pedro batendo na porta
Limpeza entrando no banheiro
Limpeza limpando o banheiro
Limpeza saindo do banheiro
Pedro entrando no banheiro
Pedro fazendo coisa demorada
Pedro dando descarga
Pedro lavando a mao
Pedro saindo do banheiro
João entrando no banheiro
João fazendo coisa rapida
João dando descarga
João lavando a mao
João saindo do banheiro
```

Repare que, depois da limpeza, o Pedro e o João conseguiram usar o banheiro! O programa terminou normalmente com todos os threads finalizados