10

Mãos à obra: Mais limpeza

Agora podemos pensar que nossos convidados são mais exigentes ainda. Assim que alguém usou o banheiro, ele fica sujo novamente, razão suficiente para outros convidados negarem o serviço!

1) Na classe `Banheiro`, no final dos métodos `fazNumero1()` e `fazNumero2()`, altere o booleano `ehSujo` para `true`:

Segue, como exemplo, o primeiro método:

```
public void fazNumero1() {

    // restante do código omitido

    this.ehSujo = true; //novo

    System.out.println(nome + " dando descarga");
    System.out.println(nome + " lavando a mao");
    System.out.println(nome + " saindo do banheiro");
}
```

Igualmente no método `fazNumero2()`.

2) Ainda na classe `Banheiro` troque o `if` com `while`. Ou seja, enquanto o banheiro está sujo, a thread fica esperando:

```
public void fazNumero1() {

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

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

    synchronized (this) {

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

        while (this.ehSujo) { //novo, trocando if com while
            esperaLaFora(nome);
        }

        // restante do código omitido
}
```

Igualmente no método `fazNumero2()`.

3) Vamos fazer que a nossa limpeza volte a verificar se o banheiro está sujo ou não. É preciso voltar e repetir o serviço!

Na classe `TarefaLimpeza` crie um `while(true)`:

```
public class TarefaLimpeza implements Runnable {
```

```
    //construtor e atributo omitido

    @Override
    public void run() {
        while(true) {
            this.banheiro.limpa();
            try {
                Thread.sleep(15000);//limpando cada 15s
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

4) Agora teste novamente no seu código. Rode a classe `Principal` e verifique se a limpeza volta a limpar o banheiro e notifica as outras threads.