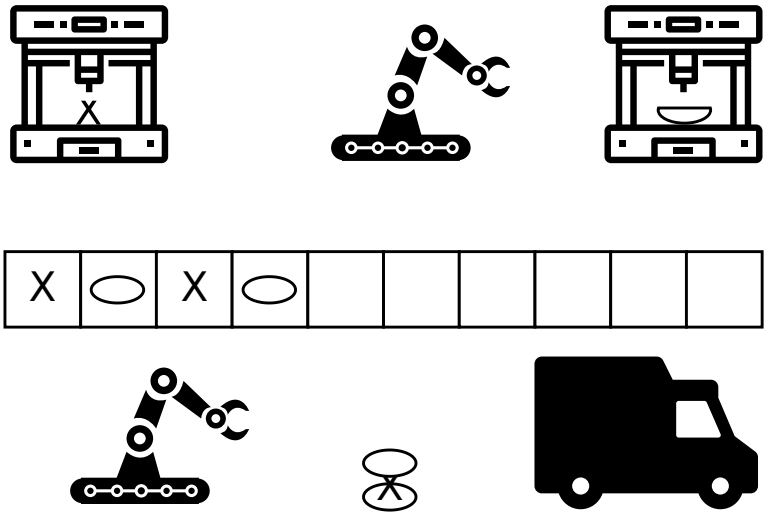
Atividade Prática - Monitores

Você deve simular o processo de fabricação de um produto utilizando threads e monitores em Java.



A fábrica em questão produz um produto que é montado usando uma peça do tipo X e duas peças do tipo O. As peças do tipo O são manufaturadas por uma impressora 3D e colocadas por um braço robótico em um buffer com `maxSize` lugares. Nesse mesmo buffer são colocadas as peças do tipo X, fabricadas por outra impressora 3D. Um robô com uma câmera acoplada ao seu braço mecânico faz a montagem do produto utilizando as peças contidas no buffer. Isso é feito colocando uma peça X sobre uma peça O e depois sobrepondo a elas outra peça O. Depois de montado, o produto é despachado para entrega.

Regras

- Impeça o acesso simultâneo dos robôs ao buffer, de modo a evitar colisões que possam danificá-los.
- O buffer deve ter no máximo `maxSize-1` peças do tipo O, caso contrário não será possível montar um produto. De forma semelhante, podem haver somente `maxSize-2` peças do tipo X no buffer. Atingidos esses limites, a impressora 3D que fabrica a peça deve ser colocada em modo de espera até que uma peça daquele tipo seja retirada do buffer.
- As duas impressoras devem ser colocadas em espera quando o buffer estiver lotado.
- O robô que faz a montagem do produto deve aguardar até que o buffer contenha pelo menos 2 peças do tipo O e 1 peça do tipo X.
- Utilize o monitor do objeto (`wait()`, `notify()` e/ou `notifyAll()`) e uma única fila (LinkedList<Piece>).

Código

- `Buffer.java`: **Implemente a lógica do buffer aqui**. Os métodos **`add(Piece)`** e **`takeOXO(List<Piece> xList, List<Piece> oList)`** devem bloquear por tempo indefinido, conforme as regras acima. O método **`takeOXO()`** deve retirar duas peças do tipo **O** e uma peça do tipo **X**, inserindo cada peça no `List<Piece>` correspondente fornecido como argumento.
- `Robot.java`: funcionamento do Robô (**já está pronto**).
- `Printer.java`: Impressora 3D (**já está pronta**).
- `Main.java`: executa a simulação da fábrica em funcionamento.
- `BufferTests.java`: Testes

Correção automática

O script de correção (**grade-monitors.sh**) está dentro do esqueleto e deve ser executado na própria pasta onde está. Você deve implementar as classes já criadas. Não crie suas soluções em novas classes ou o script corretor não as irá encontrar. Os testes estão visíveis na classe `BufferTest`. O arquivo de testes será substituído durante a avaliação.

Caso queira usar features do Java 8+ altere o pom.xml.

Entrega do Exercício

Submeta um arquivo **.tar.gz (ou zip)** na **mesma estrutura do esqueleto**dado como ponto inicial nessa tarefa. O uso do esqueleto fornecido é **obrigatório**. Utilize **make submission** para garantir que será gerado um arquivo conforme solicitado.

O prazo para entrega é **4 de Novembro às 23h55**.

Esqueleto

(Atualizado em 31/10/2018 - 16:31)

- `pom.xml` não rodava testes no Java 10 e 11

atividade_10.tar.gz
pom.xml