Laboratório de Computação Paralela e Sistemas Distribuídos – AUDI (Fábrica de Automóveis)

Autores: Frederico Oliveira Freitas
Luan Felipe Ribeiro Santos
Talles Souza Silva

1. Introdução

O objetivo deste trabalho prático/avaliação consiste em desenvolver um software para controlar a fabricação de carros da empresa AUDI com cada linha de produção sendo bem segmentada e dividida em: produção de pneus, produção de motores, produção de carrocerias, produção de bancos e produção de eletrônica embarcada.
Espera-se com a aplicação o foco em evitar a espera ocupada ainda que cada linha de produção tenha um tempo diferente

2. Implementação

Pacote Apresentação

Classe: Inicial

Classe que executa as threads produtor e consumidor

public class Inicial extends JFrame(): Definições de variáveis e componentes da tela inicial, bem como o layout que será retratado após a execução do código fonte, com a contagem do número de carros produzidos a partir da produção de pneus, produção de motores, produção de carrocerias, produção de bancos e produção de eletrônica embarcada.

public void geraMovimento(): Inicializa a thread responsável pelo movimento do carro que é colocado na carreta ao final da sua produção.

public class mudaQuantidade extends Thread(): Invoca a thread responsável pela mudança da quantidade do estoque das peças e quantidade dos carros produzidos.

public class Movimento extends Thread(): Invoca a Thread que realiza os movimentos das peças na esteira durante a produção. Caso não consiga, lança uma exceção. Verifica se o estoque está cheio e se sim, fica parada.

public class MovimentoCarro extends Thread(): Invoca a Thread que realiza o movimento do carro até a carreta ao final de sua produção.

public class MovimentoCegonha extends Thread(): Invoca a Thread que realiza o movimento da carreta após ser carregada por 10 carros. Verifica se já foram produzidos os 10 carros, se sim carregar a carreta, se não, prossegue com o carregamento.

public class produçãoHora extends Thread(): Invoca a Thread que controla o tempo de produção dos carros a partir da média de carros produzidos.

Classe Main

public static void main(String[] args): Instancia uma nova Cesta com cada elemento respectivo da produção, além de invocar a tela de funcionamento que será mostrada.
Realiza a chamada para executar as linhas de produção de pneus, motores, carrocerias, bancos e de eletrônica embarcada.
Executa o consumidor instanciando um novo CarroConsumidor, com os parâmetros de cada componente do carro sendo colocado na Cesta. A produção é iniciada com o uso do synchronized e o controle a partir do notify().

Pacote Negócio

Classe CarroConsumidor

Classe que consome os itens colocados na cesta

public CarroConsumidor(Inicial frame, Cesta cestaPneus,Cesta cestaMotor,Cesta cestaEletronica,Cesta cestaCarroceria,Cesta cestaBancos,Cesta CestaCarros)(): Método construtor para a identificação de cada elemento da produção que é colocado na Cesta.

public void run(): Método que invoca a Thread implementada por Runnable e que controla a espera ocupada através do wait() e notify(). Ela verifica a quantidade necessária para produzir um carro. Ou seja, ela é responsável por consumir a produção disposta na cesta.
A partir disso, ela tira da cesta: 1 motor, 4 pneus, 5 bancos, 1 carroceria e uma parte eletrônica.

public void init(): Método que dispara a execução da Thread.

.

Classe Cesta

Classe responsável por produzir os elementos que serão retirados pela parte consumidora.
Ela adiciona e sincroniza cada elemento da produção e os dispõe para serem consumidos pelo CarroConsumidor, retornando a quantidade disponível na cesta.

Classe LPBancos

Classe que produz os bancos e os coloca na cesta

public void run(): Método que invoca a Thread implementada por Runnable e que é responsável pela produção dos bancos que são adicionados na cesta.
Faz a utilização do wait(); afim de aguardar se a produção está ocupada. Lança uma exceção caso não seja possível a espera.

public void init(int num): Método que dispara a execução da Thread.

Classe LPCarroceria

Classe que produz as carrocerias e as coloca na cesta

public void run(): Método que invoca a Thread implementada por Runnable e que é responsável pela produção das carrocerias que são adicionadas na cesta.
Faz a utilização do wait(); afim de aguardar se a produção está ocupada. Lança uma exceção caso não seja possível a espera.

public void init(int num): Método que dispara a execução da Thread.

Classe LPEletronica

Classe que produz as peças eletrônicas e as coloca na cesta

public void run(): Método que invoca a Thread implementada por Runnable e que é responsável pela produção das peças eletrônicas que são adicionadas na cesta.
Faz a utilização do wait(); afim de aguardar se a produção está ocupada. Lança uma exceção caso não seja possível a espera.

public void init(int num): Método que dispara a execução da Thread.

Classe LPMotores

Classe que produz os motores e os coloca na cesta

public void run(): Método que invoca a Thread implementada por Runnable e que é responsável pela produção dos motores que são adicionados na cesta.
Faz a utilização do wait(); afim de aguardar se a produção está ocupada. Lança uma exceção caso não seja possível a espera.

public void init(int num): Método que dispara a execução da Thread.

Classe LPPeneus

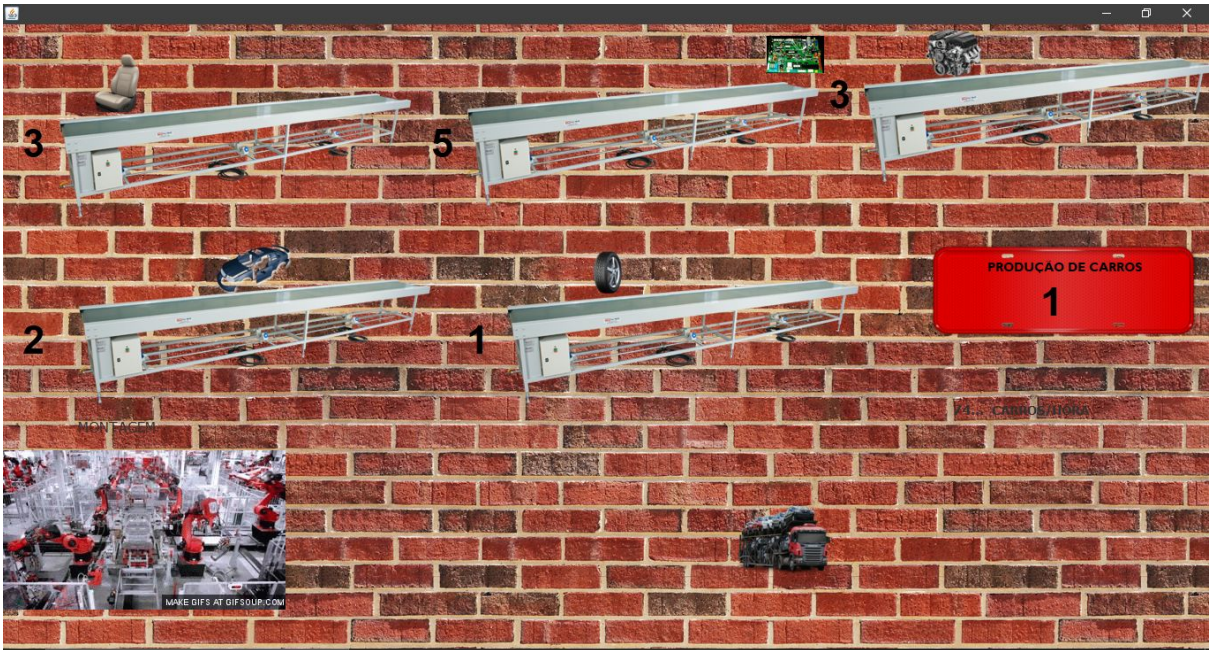
Classe que produz os pneus e os coloca na cesta

public void run(): Método que invoca a Thread implementada por Runnable e que é responsável pela produção dos pneus que são adicionados na cesta.
Faz a utilização do wait(); afim de aguardar se a produção está ocupada. Lança uma exceção caso não seja possível a espera.

public void init(int num): Método que dispara a execução da Thread.

3. Testes

- Interface gerada após a execução do código: retrata cada uma das peças passando pela esteira e sendo produzidas; bem como a quantidade de peças geradas a partir do seu tempo próprio de produção;
- Com a contagem dos carros produzidos, ao atingir a quantidade de 10 carros, a cegonheira é carregada.



4. Anexos

- Inicial.java
- Main.java
- CarroConsumidor.java
- Cesta.java
- LPBancos.java
- LPCarroceria.java
- LPEletronica.java
- LPMotores.java
- LPPneus.java