Ensamblador de partes de un producto

202010316 – Joseph Jeferson Marroquín Monroy

Resumen

Este software es una aplicación solicitada por la empresa Digital Intelligence, S.A, se implementa un algoritmo el cual consiste en leer dos archivos de entrada donde se especifica el proceso de ensamblaje de cada producto.

Por medio de este software se controla la maquina para construir cualquier producto ensamblado automáticamente los componentes que lo conforman haciendo mas eficiente el proceso de elaboración.

Palabras clave

Implementación de estructuras de datos TDA, Memoria Dinámica, XML.

Abstract

This software is an application requested by the company Digital Intelligence, S.A. It implements an algorithm which consists of reading two input files where the assembly process of each product is specified.

By means of this software the machine is controlled to construct any product assembled automatically the components that make it more efficient the production process.

Keywords

Implementation of TDA data structures, Dynamic Memory, XML.

Introducción

Desarrollamos un software que sigue un algoritmo el cual lee un archivo de entrada donde tiene instrucciones dadas para la elaboración de un producto, agregando los datos a una lista simple donde lo vamos a ir almacenando la cantidad de componentes necesarios, líneas de producción disponible, tiempo que se demora en realizar una acción el brazo robótico y la elaboración a seguir para construir determinado producto.

Para la lectura de cada paso a seguir en la elaboración usamos expresiones regulares donde buscamos la instrucción a seguir por el brazo robótico recorriendo cada línea de producción con sus respectivos componentes.

Desarrollo del tema

Una clase es un modelo para crear a partir de ella objetos, es la encargada de almacenar la información, crear características a partir del objeto, agrupándose los objetos en una misma clase.

Una función es un bloque de código reutilizable que se encarga de realizar determinada tarea.

TDA

Es un conjunto de datos u objetos al cual se asocian operaciones. Provee una interfaz con la cual es posible realizar operación permitidas.

Implementación de librerías

```
import tkinter as tk
import xml.etree.ElementTree as xml
import xml.etree.ElementTree as ET
```

```
import re, cv2
import webbrowser
import imgkit
```

Usamos xml.etree.ElementTree la cual se encarga de analizar o crear documento con extensión XML representándolos como un árbol, tambien nos ayudamos de tkinter para realizar nuestra interfaz, la librería re no ayuda para identificar la elaboracion a seguir de cada producto por medio de expresiones regulares.

Implementación de clases

```
class ListaNoOrdenada:
```

Esta clase es la encargada de ir agregando datos a nuestra lista simple o de realizar una búsqueda dentro de la misma para encontrar determinado producto.

```
class Nodo:
```

En esta clase definimos un constructor que nos ayudara a almacenar información, creamos un apuntador a nulo a cada nuevo dato que recorramos y retorna datos.

```
class Maquina:
```

Esta clase es la encargada de realizar nuestro grafo por medio graphviz donde se detalla el proceso de elaboración de cada producto.

```
class Reportehtml:
```

Esta clase se encarga de realizar todo el proceso de ensamblaje detallándolo en una tabla html generando un reporte, tambien se realiza el archivo de salida con extensión XML donde tambien se muestra los pasos a seguir para elaborar cada producto.

Implementación de funciones

```
def cargarSimulacion():
```

Esta función se encarga de leer el archivo de entrada donde se encuentran los productos que se van a simular dentro de la interfaz.

```
def cargarMaquina():
```

Esta función se encarga de leer el archivo que el usuario haya subido y cargarlo a memoria, usando la librería de Element Tree lo que nos representa el archivo como un árbol, leyendo cada dato como líneas de producción, cantidad de componentes, nombre del producto o elaboración.

```
def mostrarGrafo():
```

Esta función es la encargada de realizar el grafo de la elaboración de determinado producto guardándolo como un archivo png.

```
def abrirHTML():
```

Esta función es la encargada de generar el reporte en formato HTML donde se detalla el proceso para elaborar un producto.

```
def datosEstudiante():
```

Esta función abre una nueva ventana donde nos muestra por medio de label la información del estudiante.

```
def infoAPP():
```

Esta función abre una nueva ventana donde nos muestra por medio de un label más información acerca de la aplicación.

```
def campoTexto():
```

Esta es la función mas importante del programa ya que es la encargada de mostrar la simulación del producto escogido por el usuario mostrándolo en tiempo real en la interfaz.

```
def generarGraphviz(self):
```

Esta función reconoce el nombre del producto a simular haciendo los pasos de elaboración por medio de un grafo usando graphviz.

```
def repHtml(self,nombree,lista,lista2):
```

Esta función es la que genera el reporte de HTML recibiendo el nombre del producto a ensamblar, la cantidad de líneas de producción o los componentes que se requieren para elaborar determinado producto, al mismo tiempo generamos nuestro archivo de salida con extensión XML donde se detalla el proceso a seguir para ensamblar.

```
def agregaMatriz(self, elabo):
```

Esta función es la encargada de agregar los pasos para ensamblar un producto agregándolo según el nombre que este tenga guardándolo en una lista simple.

```
def agregar(self,item):
```

En esta función es la encargada de ir creando nuestra lista simple agregando datos según el archivo de entrada que el usuario suba.

```
def buscar(self,item):
```

Esta función es la encargada de buscar determinado producto en nuestra lista simple, nos sirve al momento de querer simular un producto en específico.

Conclusiones

Esta aplicación nos representa la conexión entre listas simplemente enlazadas y el resultado de funciones para generar información gráfica.

La implementación de un software para que realice un proceso de ensamblaje automáticamente resulta más optimo y eficiente.

Referencias bibliográficas

It.uc3m.es. 2021. *Listas enlazadas simples* . [online] Disponible en: http://www.it.uc3m.es/java/2011-12/units/pilas-colas/guides/2/guide_es_solution.html

Vega, R. (s. f.). PHAROS. https://pharos.sh/leer-y-escribir-archivos-xml-en-python-con-pandas/

Diagrama de clases

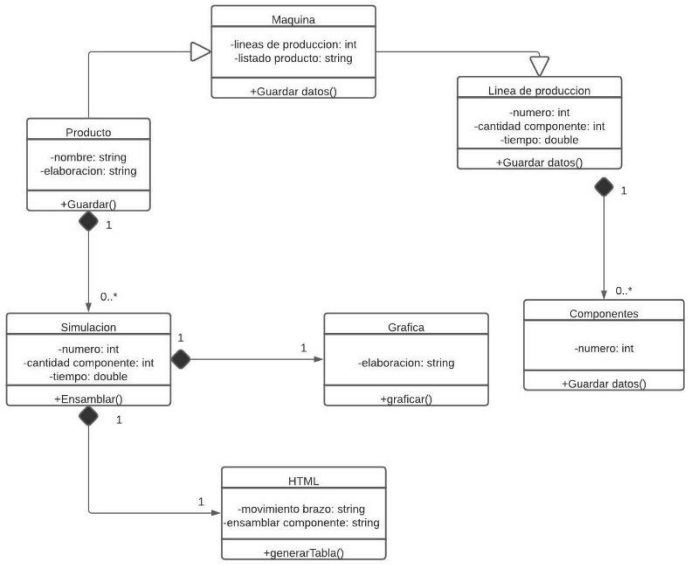

Figura 1. Diagrama de clases

Fuente: elaboración propia.