Kevyn Guadamuz Rojas

Carnet: 2017021057

Roger Valderrama

Carnet: 2017113167

Taller de Programación

Tarea Programada #3

Grupo #1

Profesor: Jeff Schmidt

Fecha de Entrega:
Domingo 18 de Junio

2017

Índice:

Introducción

Para este proyecto se pretende desarrollar un simulador de una estación de trenes del Tecnológico de Costa Rica. El simulador pretende manejar los trenes con las diferentes rutas y simular el abordaje, la salida y la llegada de los trenes

Con este proyecto se pretende aumentar los conocimientos del lenguaje que se utiliza en este curso el cuál es Python. Se trabajará con conceptos de Programación Orientada a Objetos y también con el concepto de Lista Doblemente Enlazadas. En la parte de la Programación Orientada a Objetos se utilizará para manejar cada objeto como lo son los trenes, los vagones y las maquinas.

El concepto de Listas Doblemente Enlazadas se puede observar en los vagones, con el cual cada vagón pertenece a un nodo y el tren en si es la lista completa. El simulador debe poseer un mecanismo automático que asigne los vagones a los trenes de acuerdo a la demanda de usuarios.

La simulación de la estación de tren se trabajará con una interfaz gráfica lo cual permitirá que se adquieran y se complementen los conocimientos sobre interfaz gráfica ya se la librería que se esté utilizando, en nuestro caso se utilizará Tkinter por lo que el conocimiento ya adquirido con proyectos anteriores se complementara con los conocimientos que se van a adquirir con este nuevo proyecto.

Descripción del Problema

El principal objetivo con este proyecto es desarrollar un simulador de una estación de trenes que será construida en un futuro en el Instituto Tecnológico de Costa Rica. La simulación va a presentar el funcionamiento de llegadas y salidas de trenes, pudiendo administrarse la cantidad de pasajeros, la cantidad de vagones a asignar por cada tren y otros aspectos. No se van a considerar aspectos como capacidades de la vía férrea ni tampoco seguridad en el manejo y asignación de rutas. Tampoco se van a manejar filas de pasajeros (colas) ni pagos de para adquirir los tiquetes.

El simulador debe tener una serie de botones o menús que permitan realizar las siguientes funciones:

- Iniciar simulación: lee los datos de un archivo de configuración.
- Lista de rutas por hora: para la siguiente hora muestra todos los trenes que tiene planeada una salida o llegada.
- Estimación de demanda por ruta: por medio de esta opción se genera un número aleatorio de la cantidad de personas que van a viajar en una de las rutas a manejar en la hora. Aplica solo para salidas de trenes.
- Administración de vagones: el controlador puede realizar de forma manual o automática la asignación de vagones para una ruta, de acuerdo a la demanda.
- Los vagones libres son cargados desde el archivo de configuración.
- Salida de tren: desaparece de la lista de trenes a administrar.
- Llegada de tren: se registra la llegada y se liberan los vagones.

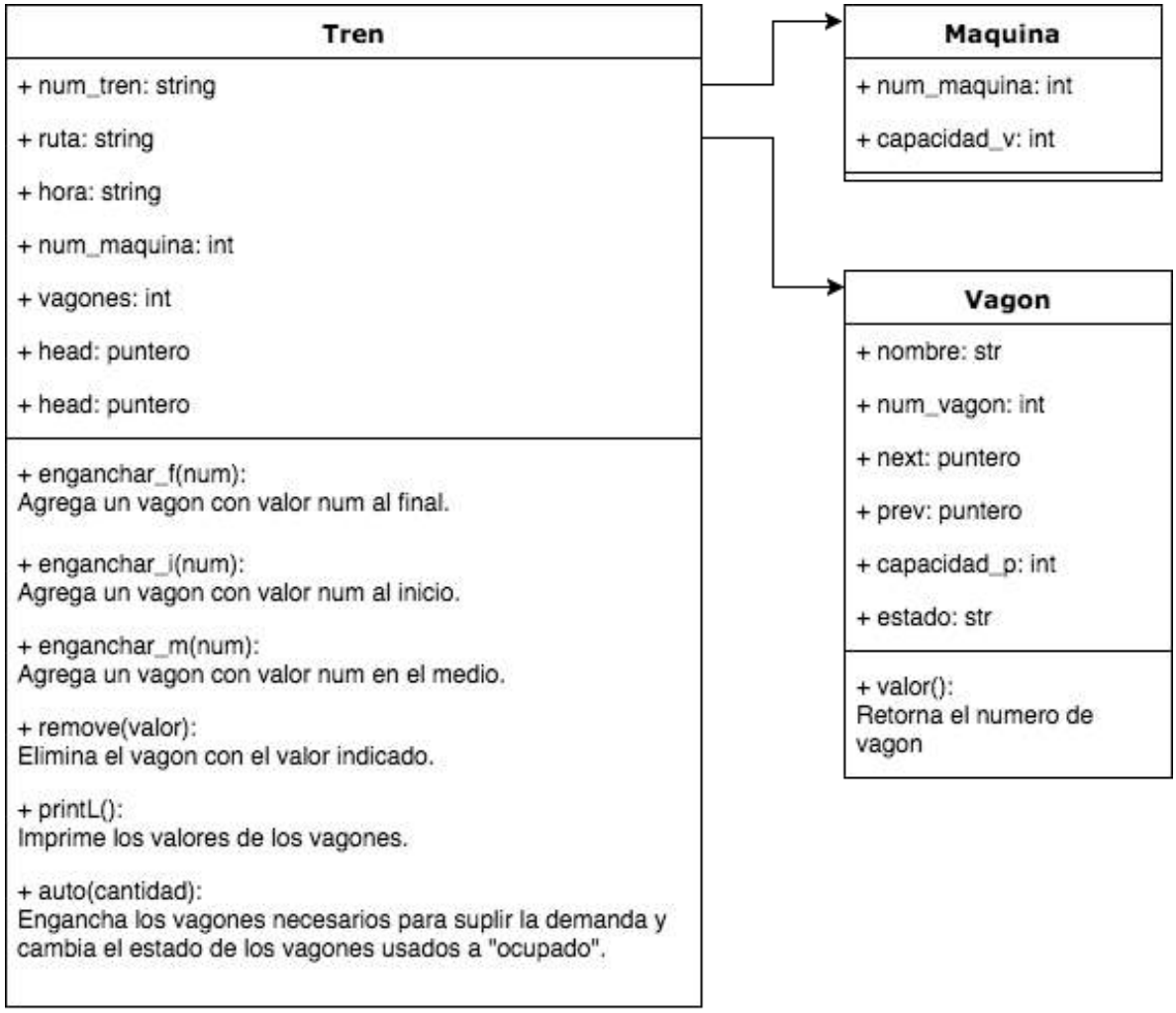
Todas estas funciones deben mostrar elementos gráficos que permitan al controlador lo que está pasando.

El tren deberá ser implementado en programación orientada a objetos. Se deberá definir los métodos y atributos de cada objeto. Pueden existir algunos métodos que se requieran y no estén descritos en la presente definición. Pueden realizarse modificaciones al modelo de objetos enunciado, siempre y cuando las mismas se documenten.

Para mejorar la calidad y presentación de la tarea, debe investigarse el uso de algunas funciones referentes a validaciones de datos y despliegue de información. Las funciones que podrían utilizarse, entre otras son:

- Utilización de multimedia: integración de animaciones, sonidos y otros.
- Generación de números aleatorios
- Manejo de archivos de texto

Diagrama de Clases:



Dificultades Encontradas

Lectura y escritura de archivos de texto:

El primer problema que apareció en el transcurso del desarrollo del proyecto es la lectura y la escritura de archivos de texto, los cuales deben utilizarse para inicializar las instancias de cada objeto.

Para la solución de este problema se implementa el uso de archivos ".cvs" los cuales son archivos que los datos se encuentran separados por comas.
Para la lectura de estos archivos se importa la libre CSV que trae Python y se utiliza esta librería para el manejo de los archivos ".csv", los cuales pueden ser modificados por medio del bloc de notas.

Definición de la función que administre los vagones automáticamente:

Al implementar la lógica de esta función nos vimos en un problema ya que no encontrábamos la forma de ver como leer el estado del vagón (si se encontraba libre u ocupado) y ver la capacidad que este poseía. Y encontrar la manera que esta función cuando la demanda de pasajeros ya fuese cubierta.

Con la lectura de los archivos ".csv" se inicializa las instancias y desde ahí se empieza a trabajar con las instancias. De manera que existe una función que reciba cuales vagones se encuentran en estado libre y crea una lista con las instancias que poseen ese estado, luego se envía a la función que realiza el enganche de los vagones de manera automática y cuando algún vagón se engancha este cambia su estado a ocupado.

Definición de la función que administre los vagones de manera manual:

Se necesitaban una función que permitiera desde la interfaz gráfica realizar el acomodo de vagones de forma manual, permitiendo así elegir cualquier vagón que se encontrara en estado "Libre". También permitiendo que el vagón pueda salir de una forma cuando ya se administraran los vagones.

Se logra implementar la forma de que la función que permita la administración de los vagones de manera manual. Se implementa con un conjunto de funciones que realizan diversas tareas, por ejemplo: la verificación de los trenes seleccionados, la verificación de los vagones a utilizar y estas llaman a otras funciones para que se realice el enganche de los vagones.

Diseño de la interfaz:

Se busca crear una interfaz agradable a la vista para los usuarios finales. De manera que se implementan varias pruebas de interfaces para ver cuál era la escogida.

Se selecciona una interfaz que al gusto de los integrantes del grupo es agradable a la vista.

Actualización de Etiquetas:

En la interfaz se quería mostrar la lista con los trenes disponibles. De modo que en un solo Label se pudiera implementar la vista de los strings de una lista.

Se logra encontrar la solución implementando el método ".join" para particionar la lista en sublistas y que de esta manera se pueda mostrar sin implementar una función que realice este particionamiento, Se implementa la actualización de las etiquetas pero estás se actualizan de manera atrasada.

Sincronización en tiempo real la llegada de los trenes:

Implementar la manera de que la llegada de los trenes se sincronizara con el reloj tiempo real del sistema para que a la hora preestablecida en los archivos CSV se diera la llegada del tren y con ella la liberación del tren en uso y los vagones.

No se logra implementar esta función, por lo que se decide, por falta de tiempo, implementar un botón que reinicialice todos los estados de las instancias a un estado "Libre".

Verificación del número correcto de vagones de acuerdo a la demanda:

Al usar el modo manual se debe verificar que la cantidad de vagones sea la correcta para satisfacer la demanda que se tiene en ese momento que es generado con una función que genera números aleatorios.

No se logra implementar tal función.

Problemas con las imágenes:

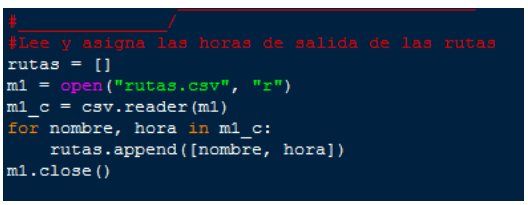
Las imágenes presentaron un inconveniente ya que no se encontraban el tamaño ideal de ellas. Los botones también no se encontraban de manera correcta o los suficientes. Los vagones presentaron el mismo inconveniente ya que o eran muy grande o eran muy pequeños.

Se soluciona editando las imágenes a prueba y error hasta que estas tomaran un tamaño adecuado, los botones al no encontrar varios de diferentes colores se utilizó ilustrador para cambiar los colores de los botones.

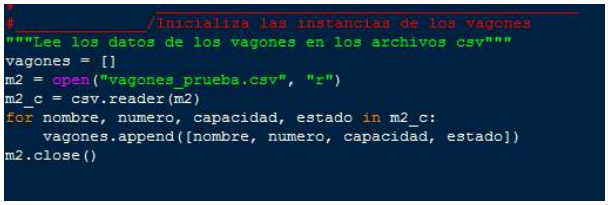
Análisis de Resultados

Lectura y escritura de archivos de texto:

Se importa el módulo CSV que permite manejar los archivos de texto de una manera más sencilla y estos contienen los datos para que sean inicializadas las instancias.



```
#_____/
#Lee y asigna las horas de salida de las rutas
rutas = []
m1 = open("rutas.csv", "r")
m1_c = csv.reader(m1)
for nombre, hora in m1_c:
    rutas.append([nombre, hora])
m1.close()
```



```
#_____/Inicializa las instancias de los vagones
"""Lee los datos de los vagones en los archivos csv"""
vagones = []
m2 = open("vagones_prueba.csv", "r")
m2_c = csv.reader(m2)
for nombre, numero, capacidad, estado in m2_c:
    vagones.append([nombre, numero, capacidad, estado])
m2.close()
```

```
a.start()"""
#-------------------

"""x = Tren("thomas","s
x.printL()
x.auto(100)
x.printL()
b = Tren("Ricardo","s-c
b.auto(100)
b.printL()
a = Tren("Lola", "SS",
a.auto(600)"""
print(print_nombres(vag
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.19
 on win32
Type "copyright", "credits" or "license()" for more informatio
>>>
======= RESTART: C:\Users\Extreme PC\Documents\GitHub\tren-tec
[['vagon1', '50'], ['vagon2', '40'], ['vagon3', '100'], ['vago
n5', '60'], ['vagon6', '100'], ['vagon7', '50'], ['vagon8', '4
0'], ['vagon10', '100']]
None
```

Definición de la función que administre los vagones automáticamente:

Se implementa dos funciones para que sirva correctamente este método. Las cuales son “vagon_libre”, lo que realiza esta función es que revisa el estado de los vagones y retorna

una lista con todos los vagones que tienen el estado “Libre” y la otra función “auto” realiza el enganche de los vagones hasta que la demanda de los usuarios sea cubierta.

```
#_________/Función que realiza una lista con los
def vagon_libre(lista):
    vagones_libres = []
    for i in range(len(lista)):
        temp = lista[i]
        if temp.estado == 'Libre':
            vagones_libres += [temp]
    return vagones_libres
#####----------------------------------#####
```

```
#_____________/Funcion que ejecuta lo automatico(No term
def auto(self, cantidad):  # Función para los vagones automa
    temp = vagon_libre(vagones_a_evaluar)
    i = 0
    conta = 0
    prueba = []
    while conta <= cantidad:
        if i == len(temp):
            print("Cantidad de vagones insuficientes")
            break
        elif self.vagones == 0:
            self.enganchar_i(temp[i].num_vagon)
            temp[i].estado = "Ocupado"
            prueba += [temp[i].num_vagon]
            conta += int(temp[i].capacidad_p)
            i += 1


        else:
            self.enganchar_f(temp[i].num_vagon)
            temp[i].estado = "Ocupado"
            prueba += [temp[i].num_vagon]
            conta += int(temp[i].capacidad_p)
            i += 1
```

Definición de la función que administre los vagones de manera manual:

Se implementa una función que verifica los trenes y los vagones seleccionados.

```
"""Funcion que verifica el comando del boton enganchar al inicio"
def verifica_manual_inicio():
    rm = radio_manual.get()
    a = "\n".join(map(str, print_nombres(vagones_a_evaluar)))
    if rm == 1 and tren5.estado == "Libre":#Verifica si el tren1
        if shell.get() == "vagon1"and vagon1.estado == "Libre":
            tren5.enganchar_i(vagon1.num_vagon)
            vagon1.estado = "Ocupado"
            vagones_lbl.config(text=a)
            ventana_principal.update()
        elif shell.get() == "vagon2"and vagon2.estado == "Libre":
            tren5.enganchar_i(vagon2.num_vagon)
            vagon2.estado = "Ocupado"
            vagones_lbl.config(text=a)
```

```
"""Funcion que verifica el comando del boton enganchar al medio"""
def verifica_manual_medio():
    rm = radio_manual.get()
    a = "\n".join(map(str, print_nombres(vagones_a_evaluar)))
    if rm == 1 and tren5.estado == "Libre":#Verifica si el tren1 se encuentra en la
        if shell.get() == "vagon1"and vagon1.estado == "Libre":
            tren5.enganchar_m(vagon1.num_vagon)
            vagon1.estado = "Ocupado"
            vagones_lbl.config(text=a)
            ventana_principal.update()
        elif shell.get() == "vagon2"and vagon2.estado == "Libre":
            tren5.enganchar_m(vagon2.num_vagon)
            vagon2.estado = "Ocupado"
            vagones_lbl.config(text=a)
            ventana_principal.update()
        elif shell.get() == "vagon3"and vagon3.estado == "Libre":
            tren5.enganchar_m(vagon3.num_vagon)
            vagon3.estado = "Ocupado"
            vagones_lbl.config(text=a)
            ventana_principal.update()
```

Diseño de la interfaz:

Se implementa una interfaz que se considera agradable a la vista.

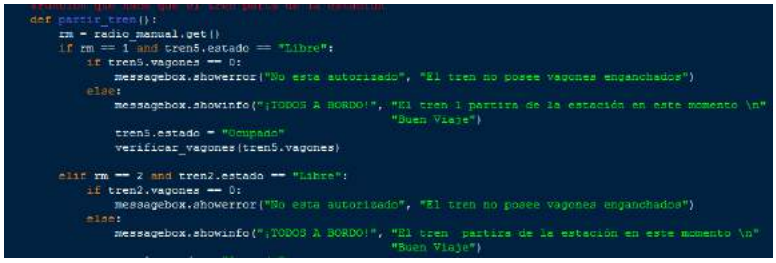
Actualización de Etiquetas:

Para la actualización de las etiquetas se realiza un .update del Label con una nueva configuración de texto.

```
def evalua():
    r = radio.get()
    a = "\n".join(map(str, print_nombres
    if r == 1 and tren5.estado == "Libre"
        tren5.auto(demanda_variable.get()
        messagebox.showinfo("Los vagones
                            str(tren5.pri
        tren5.printL()
        tren5.estado = "Ocupado"
        vagones_lbl.config(text=a)
        ventana_principal.update()
```

Verificación del número correcto de vagones de acuerdo a la demanda:

Debido a que no se logra implementar de manera adecuada la función que permita verificar que la cantidad de vagones cubre correctamente la demanda. Se implementa un función que realice una animación y cambie el estado del tren y de los vagones.



```
def partir_tren():
    rm = radio_manual.get()
    if rm == 1 and tren5.estado == "Libre":
        if tren5.vagones == 0:
            messagebox.showerror("No esta autorizado", "El tren no posee vagones enganchados")
        else:
            messagebox.showinfo(";TODOS A BORDO!", "El tren 1 partira de la estación en este momento \n"
                                                   "Buen Viaje")
            tren5.estado = "Ocupado"
            verificar_vagones(tren5.vagones)

    elif rm == 2 and tren2.estado == "Libre":
        if tren2.vagones == 0:
            messagebox.showerror("No esta autorizado", "El tren no posee vagones enganchados")
        else:
            messagebox.showinfo(";TODOS A BORDO!", "El tren  partira de la estación en este momento \n"
                                                   "Buen Viaje")
```

Bitácora de Actividades

Viernes 9 de Junio:

Implementación de POO.
Se inicia con la definición de cada uno de los métodos de las diferentes clases. Kevyn Guadamuz Rojas (3 horas)

Creacion de Esqueleto de las clases Maquina, Tren, Vagon asi como varios metedos de Tren. Roger Valderrama (4 horas)

Sabado 10 de Junio:

Se investigo sobre csv y como manejar datos en archivos como si fueran una base de datos, se arreglaron los metodos enganchar inicial, final y medio y se creo un print. Roger Valderrama (8 horas)

Miércoles 14 de Junio:

Se empieza a trabajar con una interfaz de prueba.
Se empieza a trabajar con una interfaz de prueba para verificar que ningún método afectara el funcionamiento de ambas partes.
Kevyn Guadamuz Rojas (2 horas).

Implementación de Archivos CSV.
Se investiga como leer y escribir de una manera más sencilla archivos de texto y se comienza a implementar una manera de leer los archivos.
Kevyn Guadamuz Rojas (3 horas).

Se empieza a trabajar en la función automático.
Se empieza a trabajar con la lógica de la función que realiza de manera automática el enganche de los vagones, pero no funciona de manera correcta ya que no se implementa la inicialización de cada instancia de la clase Vagón.
Kevyn Guadamuz Rojas (4 horas).

Jueves 15 de Junio:

Se investiga otra forma de manejar archivos ".csv"
Se implementa otra manera de manejar los archivos ".csv" de una forma más sencilla la cual es implementando el módulo CSV para el manejo de estos archivos.
Kevyn Guadamuz Rojas (2 horas)

Se avanza la interfaz y se buscan imágenes de la estación y se editan en ilustrador, ademas de el dibujo de GAM y las vías del tren. Ademas se investiga sobre como sortear un csv.
Roger Valderrama (7 horas)

Viernes 16 de Junio:

Inicialización de Instancias de la clase Vagón.
Se implementa como inicializar cada instancia de Vagón desde el archivo de texto.
Kevyn Guadamuz Rojas (1 hora)

Función "vagon_libre".
Se mejora la función vagón libre para que esta trabaje con las instancias que ya se crean desde el archivo ".csv"
Kevyn Guadamuz Rojas (2 horas)

Función "auto".
Se trabaja en mejorar la función auto para que esta trabaje de la misma forma con las instancias con el estado libre provenientes de la función "vagon_libre" y que este enganche de manera correcta y cambie el estado del vagón a "Ocupado".
Kevyn Guadamuz Rojas (3 horas)

Asignación de Variables en las rutas.
Se implementa la manera de que la ruta sea leída desde el ".csv" para que pueda ser mostrada en la interfaz
Kevyn Guadamuz Rojas (30 minutos)

Se avanza con la interfaz y se crean 2 ventanas nuevas (Manual e Automatico) y se conseguen varios dibujos los cuales edite con ilustrador para que representen los trenes y vagones. Roger Valderrama (8 horas)

Sábado 17 de Junio:

Documentación Externa.
Se inicia con la documentación externa del proyecto. Kevyn Guadamuz Rojas (2 horas)

Instancias de la clase Tren y Maquina.
Se cargan las instancias de la clase máquina y tren. Kevyn Guadamuz (30 minutos)

Se empieza a juntar la parte lógica con la interfaz.
Se tenía la interfaz gráfica y la parte lógica en dos archivos .py distintos se empieza a juntar y asignar las funciones correspondientes a los botones. Kevyn Guadamuz Rojas (5 horas)

Actualización de Etiquetas.
Se implementa la actualización de las etiquetas de que posee los vagones libres.
Kevyn Guadamuz Rojas (1 horas)

Se trabaja con la función auto.
Se implementa la función auto de manera gráfica y unirla con la parte lógica del programa.
Kevyn Guadamuz (2 horas)

Se crean las animaciones de los trenes. Roger Valderrama (5 horas)

Domingo 18 de Junio:

Se afinan detalles en cuanto a la parte grafica con la lógica y se finaliza la interfaz. Ambos (12 horas)

Se termina la Documentacion externa. Ambos (2 horas)

Estadística de Tiempos

FUNCION	Kevyn Guadamuz	Roger Valderrama	TOTAL
Análisis de requerimientos	12 horas	12 horas	24 horas
Diseño	10 horas	13 horas	23 horas
Investigación de funciones	5 horas	8 horas	13 horas
Programación	20 horas	15 horas	35 horas
Documentación interna	2 horas	2 horas	4 horas
Pruebas	3 horas	4 horas	7 horas
Elaboración documento	5 horas	2 horas	7 horas
TOTAL	54 horas	56 horas	113 horas

Conclusión

Kevyn Guadamuz:

Gracias a este proyecto se amplió el conocimiento en conceptos de Memoria Dinámica y la programación Orientada a Objetos, ya que era un tema que no manejaba muy bien, con este trabajo se fortalece la parte de programación orientada a objetos. Se amplía el conocimiento en el manejo de la librería grafica Tkinter como lo son los "RadioButtons" que son widgets que no había utilizado.

Se mejora la capacidad del trabajo en equipo ya que esto es una cualidad que todo profesional necesita para la vida laboral.

Se desarrolla la parte lógica de la mente tratando de solucionar cada uno de los problemas que se fueron presentando con el desarrollo del proyecto.
El manejo de archivos de texto, para guardar de manera predeterminada las instancias de cada uno de los objetos.

Róger Valderrama:

Gracias a este proyecto practique y mejore mi comprension en los temas de memoria dinamica y Programacion Orientada a Objetos. Ademas, obtuve una mejor idea de cómo funciona una estacion de trenes y como se administra. Tambien se fortalecieron mucho mis habilidades en Diseno Grafico con la interfaz y la habilidad de trabajar en equipo, la cual sera determinante en un futuro.