Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Lenguajes Formales y de Programación

Sección B+

Fecha: 19/08/2020

MANUAL TÉCNICO

Nombre: Carné:

Gerson Rubén Quiroa del Cid 2020 00166

Índice

Requisitos del sistema

Para la instalación del videojuego, su computadora y/u ordenador debe cumplir como mínimo los siguientes requerimientos:

Para Windows

- Windows Vista SP2 (8u51 y superiores)
- Windows Server 2008 R2 SP1 (64 bits)
- Windows Server 2012 y 2012 R2 (64 bits)
- RAM: 128 MB
- Espacio en disco: 124 MB para JRE; 2 MB para Java Update
- Procesador: Mínimo Pentium 2 a 266 MHz
- Exploradores: Internet Explorer 9 y superior, Firefox

Mac OS

- Mac con Intel que ejecuta Mac OS X 10.8.3+, 10.9+
- Privilegios de administrador para la instalación
- Explorador de 64 bits
- Se requiere un explorador

Linux

- Oracle Linux 7.x (64 bits)2(8u20 y superiores)
- Red Hat Enterprise Linux 7.x (64 bits)2(8u20 y superiores)
- Suse Linux Enterprise Server 12.x (64 bits)2(8u31 y superiores)
- Ubuntu Linux 12.04 LTS, 13.x
- Ubuntu Linux 15.10 (8u65 y superiores)

IMPORTANTE: Independientemente del sistema operativo del usuario, es necesario que previamente instale Python en su computadora.

Editor utilizado: Visual Studio Code IDE 8.2

Versión de Python utilizado: 3.9.4

Introducción

El manual técnico tiene como finalidad de explicar el funcionamiento de cada función utilizado en el presente programa, con el objetivo de que no haya confusión al leer el código del programa y cualquier persona con conocimiento básico de programación en Python pueda comprender lo que se hizo en cada línea de código. Por lo cual se recomienda que toda persona que quiera leer o modificar el código vea este manual para ahorraste tiempo y así sea más fácil su comprensión.

Clase Menu (main)

Esta clase es la principal del programa donde inicialmente se ejecutará el menú con las diferentes opciones que puede elegir el usuario, dependiendo de la opción que se elija el usuario se ejecutarán las diferentes funciones detalladas más adelante.

```
152  # menú principal
153  # variables globales
154  alumnosL = list()
155  curso = Curso("", [], 0, 0, 0, 0, 0)
156
157  while(True):
158      print("====================Bienvenido!====================")
159      print("¿Qué acción desea realizar?")
160      print("1. Cargar archivo")
161      print("2. Mostrar reporte")
162      print("3. Exportar reporte")
163      print("4. Salir")
164      try:  # opciones del menú
165          menu = int(input('Ingrese un número:\n'))
166          print("\n")
167          if(menu == 1):
168              contenido = cargarArchivo()
169              if(contenido != ""):
170                  analizarArchivo(contenido)
171          elif(menu == 2):
172              mostrarReporte()
173          elif(menu == 3):
174              reportes(curso, alumnosL)
175              print("Reporte creado! :D")
176          elif(menu == 4):
177              exit()
178          else:
179              print("Opción fuera de rango\n\n")
180      except ValueError:  # si no se ingresa un número, salta la excepción
181          print("Ingrese un número.\n\n")
```

def cargarArchivo()

Esta función le pedirá al usuario que elija un archivo con extensión .lfp y posteriormente empezará a leer dicho archivo, todo el contenido leído se almacenará en la variable “contenido”, la función retornará dicha variable.

```python
def cargarArchivo():  # carga el archivo que ingrese el usuario
    print("==================Cargar Archivo==================")
    file = askopenfilename()
    archivo = open(file, 'r')
    contenido = ""
    if(archivo.name[-4:] == ".lfp"):
        contenido = archivo.read()
        print("Archivos cargados! :D")
    else:
        print("Solo se admiten archivos con extensión .lfp")
    archivo.close()
    return contenido
```

def analizarArhivo(contenido)

Esta función se ejecutará cuando la variable “contenido” tenga información, la función tomará esta variable y con la función Split separará cada dato importante que el programa reconozca y los guardará en strings o listas de objetos.

```python
def analizarArchivo(contenido):  # analiza y separa los datos del archivo
    tituloCuerpo = contenido.split("=")  # [0] titulo, [1] cuerpo
    nombreCurso = tituloCuerpo[0].strip()  # nombre del curso
    curso.nombreCurso = nombreCurso
    alumnosParam = tituloCuerpo[1].split("}")  # [0] alumnos, [1] parámetros
    alumnosStr = alumnosParam[0].replace("{", "")  # alumnos como string
    alumnos = alumnosStr.split(",")  # lista de alumnos
    for alumno in alumnos:  # separando cada alumno
        alumno = alumno.replace("<", "")
        alumno = alumno.replace(">", "")
        alumno = alumno.strip()
        datos = alumno.split(";")  # [0] nombre, [1] nota
        nombre = datos[0].replace('"', '')
        nota = int(datos[1].strip())
        # agregando a una lista de objetos cada
        alumnosL.append(Alumno(nombre, nota))
    alumnosParam[1] = alumnosParam[1].replace(" ", "")
    parametros = alumnosParam[1].split(",")  # lista de parámetros
    curso.parametros = parametros
    ejecutarContenido(nombreCurso, parametros, alumnosL)
```

def ejecutarContenido(parametros, alumnosL)

Esta función se encargará de ejecutar los parámetros que contenga el archivo, si el archivo contiene “ASC” se ejecutará la función bubbleSortASC, si contiene “DESC” se ejecutará la función bubbleSortDESC, etc.

```python
# ejecuta los parametros del archivo
def ejecutarContenido(parametros, alumnosL):
    if("ASC" in parametros):
        bubbleSortASC(alumnosL)
    if("DESC" in parametros):
        bubbleSortDESC(alumnosL)
    if("AVG" in parametros):
        promedioF(alumnosL)
    if("MIN" in parametros):
        if("ASC" in parametros):
            min = alumnosL[0]
            curso.min = min
        else:
            min = alumnosL[-1]
            curso.min = min
    if("MAX" in parametros):
        if("ASC" in parametros):
            max = alumnosL[-1]
            curso.max = max
        else:
            max = alumnosL[0]
            curso.max = max
    if("APR" in parametros or "REP" in parametros):
        aprobadosReprobados(alumnosL)
```

def bubbleSortASC(alumnosL) y def bubbleSortDESC(alumnosL)

Estas dos funciones hacen casi que el mismo trabajo, el ordenamiento de una lista de objetos; con la única diferencia que en la primera función se ordena ascendentemente, mientras que en el segundo se ordena descendientemente.

```python
def bubbleSortASC(alumnosL):  # algoritmo de ordenamiento ascendente
    n = len(alumnosL)
    # itera cada elemento de la lista
    for i in range(n):
        swapped = False
        for j in range(0, n-i-1):
            # recorre la lista de 0 a n-i-1.
            # intercambia el elemento actual si
            # es mayor que el siguiente elemento.
            if alumnosL[j].nota > alumnosL[j+1].nota:
                alumnosL[j], alumnosL[j+1] = alumnosL[j+1], alumnosL[j]
                swapped = True
        # Si los elementos no se intercambian por el loop
        # entonces se para con un break
        if swapped == False:
            break
```

```python
def bubbleSortDESC(alumnosL):  # algoritmo de ordenamiento descendiente
    n = len(alumnosL)
    # itera cada elemento de la lista
    for i in range(n):
        swapped = False
        for j in range(0, n-i-1):
            # recorre la lista de 0 a n-i-1.
            # intercambia el elemento actual si
            # es mayor que el siguiente elemento.
            if alumnosL[j].nota < alumnosL[j+1].nota:
                alumnosL[j], alumnosL[j+1] = alumnosL[j+1], alumnosL[j]
                swapped = True
        # Si los elementos no se intercambian por el loop
        # entonces se para con un break
        if swapped == False:
            break
```

def promedioF(alumnosL)

Esta función se encarga de sacar el promedio de las notas de los alumnos del curso actual.

```python
def promedioF(alumnosL):  # promedio de las notas del curso
    suma = 0
    for alumno in alumnosL:
        suma += alumno.nota
    promedio = suma/len(alumnosL)
    curso.promedio = promedio
```

def aprobadosReprobados(alumnosL)

Esta función calcula el número de estudiantes que aprueban el curso y los almacena en la variable aprobados, y el número de estudiantes que no apruebas dicho curso que se almacena en la variable aprobados.

```python
def aprobadosReprobados(alumnosL):  # número de alumnos aprobados y reprobados
    aprobados = 0
    reprobados = 0
    for alumno in alumnosL:
        if(alumno.nota >= 61):
            aprobados += 1
        else:
            reprobados += 1
    curso.aprobado = aprobados
    curso.reprobado = reprobados
```

def mostrarReporte()

Esta función lo que hace es mostrar un reporte en consola, con todos los datos del archivo, mostrando a los alumnos en el orden deseado y los parámetros establecidos en el archivo.

```
125  def mostrarReporte(): # función para mostrar el reporte en consola
126      nombreCurso = curso.nombreCurso
127      print("=================== REPORTE ===================")
128      print("Nombre del curso: " + nombreCurso)
129      print("Alumnos: " + str(len(alumnosL)))
130      print("")
131      for alumno in alumnosL:
132          print("Nombre: " + alumno.nombre + "\nNota: " + str(alumno.nota))
133      parametros = curso.parametros
134      print("")
135      if("AVG" in parametros):
136          print("Promedio: " + str(curso.promedio))
137          print("")
138      if("MIN" in parametros):
139          print("Nota mínima: " + "\nNombre: " + curso.min.nombre + "Nota: " + str(curso.min.nota))
140          print("")
141      if("MAX" in parametros):
142          print("Nota máxima: " + "\nNombre: " + curso.max.nombre + "Nota: " + str(curso.max.nota))
143          print("")
144      if("APR" in parametros):
145          print("Alumnos aprobados: " + str(curso.aprobado))
146          print("")
147      if("REP" in parametros):
148          print("Alumnos reprobados:" + str(curso.reprobado))
149          print("")
```

Variables globales

Las únicas dos variables utilizadas en este programa fueron una lista de objetos de tipo Alumno y un objeto de tipo Curso.

```
152  # menú principal
153  # variables globales
154  alumnosL = list()
155  curso = Curso("", [], 0, 0, 0, 0, 0)
```

Clase Alumno

Esta clase es simple, solo definimos el constructor con los atributos de dicha clase.

```
1  class Alumno(object):
2      def __init__(self, nombre, nota):
3          self.nombre = nombre
4          self.nota = nota
```

Clase Curso

Esta clase, como la anterior es solamente el constructor con los atributos de dicha clase, para manejar la información en una sola clase.

```python
class Curso(object):
    def __init__(self, nombreCurso, parametros, promedio, min, max, aprobado, reprobado):
        self.nombreCurso = nombreCurso
        self.parametros = parametros
        self.promedio = promedio
        self.min = min
        self.max = max
        self.aprobado = aprobado
        self.reprobado = reprobado
```

Clase Reportes

En esta clase está el esqueleto de los reportes que se generará html cuando el usuario elija la opción de exportar reporte, dependiendo de los diferentes parámetros se generará con diferentes datos el html.

```python
def reportes(curso, alumnosL): #generación de reportes
    html = open("reporte.html", 'w')
    html.write("<!DOCTYPE html>"
               + "<html lang=\"en\">"
               + "<head>"
               + "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\">"
               + "<meta name=\"name\" content=\"Reporte\"><!--nombre de la pagina-->"
               + "<meta name=\"description\" content=\"name\"><!--autor de la pagina-->"
               + "<meta name=\"keywods\" content=\"uno,dos,tres\"><!--Palabras claavez para, separadas por comas-->"
               + "<meta name=\"robots\" content=\"Index, Follow\"><!--Mejora la busqueda-->"
               + "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\"><!--visibilidaad en diferentes pantallas -->"
               + "<link rel=\"stylesheet\" type=\"text/css\" href=\"css/styles.css\"/><!--css /estilo/tipo/ruta relativa -->"
               + "<title>Reporte</title><!--Titulo visible de la pagina-->"
               + "</head>"
               + "<body>")

    html.write("<center><!--centra todos lo que este dentro-->"
               + f"<h6 class=\"titulos\"><b> {curso.nombreCurso} </b></h6>"
               + "<!----tabla 2-->"
               + "<table class=\"steelBlueCols\">"
               + "<thead>"
               + "<tr> <th>Nombre</th> <th>Nota</th> </tr>"
               + "</thead>"
               + "<tbody>")
    for alumno in alumnosL:
        if(alumno.nota >=61):
            html.write(f"<tr> <td>{alumno.nombre}</td> <td style=\"color: blue\">{str(alumno.nota)}</td> </tr>")
        else:
            html.write(f"<tr> <td>{alumno.nombre}</td> <td style=\"color: red\">{str(alumno.nota)}</td> </tr>")
    html.write("</tbody>"
               + "</table>")
    html.write("<br>  <br>  <br>"
```

En resumen, estas son las clases y funciones en que se compone el presente programa.