<Ziqi Yan>

<2022/3/27>

<IT FDN 110>

<Assignment\_09>

Assignment 09

# Introduction

The Assignment 09 is about OOP module programming. The classes are separated into different files. And I need to implement the code by integrate each other classes by importing the corresponding modules.

# Topic\_1 Implementation of DataClass.py

In last week’s assignment, I didn’t make property setters. So this time when I’m completing DataClass.py, I set up the setter and the properties with property decorator. For get\_record and \_\_str\_\_ function, I just simply print three properties out, the only difference between these two functions is that they used different separators.

Text

Description automatically generated

Figure 1 Track class, the init constructor and properties definition with setter

For implementation of CD class, I just append track object to its cd\_tracks list. And sort it by using \_\_sort\_tracks function.

# Topic\_2 ProcessingClasses

In this Class, all the content are learned from previous classes. For select\_cd, I just use a for loop to go through every cd in the table, check if there’s one cd’s id is equal to the cd\_idx. And return it. For add\_CD and add\_Track, just simply appending them to the list/object.

Text

Description automatically generated

Figure 2 select\_cd in ProcessingClasses

# Topic\_3 IOClass

I’ve spent a lot of time on this class because I can’t figure out how to save tracks’ relations with CD into an independent file. Then it occurs to me that I can add a cd\_id element for every line of tracks. Then when I read the file, I just need to select the cd with the cd\_id, then add the track to it.

Text

Description automatically generated

Figure 3 IO class, save\_inventory

# Result

I run the TestHarness in terminal. And It was succeed:

Text

Description automatically generated

Figure 4 Test output

# Summery

In this assignment. I practiced more on OOP programming and had a better understanding of it. I made up for the properties’ definitions and setters. And improved my document with results screenshots.

# Appendix

DataProcessor

class DataProcessor:

"""Processing the data in the application"""

@staticmethod

def add\_CD(CDInfo, table):

"""function to add CD info in CDinfo to the inventory table.

Args:

CDInfo (tuple): Holds information (ID, CD Title, CD Artist) to be added to inventory.

table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.

Returns:

None.

"""

cdId, title, artist = CDInfo

try:

cdId = int(cdId)

except:

raise Exception('ID must be an Integer!')

row = DC.CD(cdId, title, artist)

table.append(row)

@staticmethod

def select\_cd(table: list, cd\_idx: int) -> DC.CD:

"""selects a CD object out of table that has the ID cd\_idx

Args:

table (list): Inventory list of CD objects.

cd\_idx (int): id of CD object to return

Raises:

Exception: If id is not in list.

Returns:

row (DC.CD): CD object that matches cd\_idx

"""

flag = False

for cd in table:

if cd\_idx == cd.cd\_id :

flag = True

return cd

if not flag:

raise Exception("id is not in the list")

@staticmethod

def add\_track(track\_info: tuple, cd: DC.CD) -> None:

"""adds a Track object with attributes in track\_info to cd

Args:

track\_info (tuple): Tuple containing track info (position, title, Length).

cd (DC.CD): cd object the tarck gets added to.

Raises:

Exception: DESCraised in case position is not an integer.

Returns:

None: DESCRIPTION.

"""

position, title, length = track\_info

try:

int(position)

except Exception as e:

raise Exception("DESCraised: position of track is zero")

cd.add\_track(DC.Track(position, title, length))

Track

class Track():

"""Stores Data about a single Track:

properties:

position: (int) with Track position on CD / Album

title: (str) with Track title

length: (str) with length / playtime of Track

methods:

get\_record() -> (str)

"""

def \_\_init\_\_(self, position, title, length) -> None:

try:

self.\_\_track\_position = int(position)

self.\_\_track\_title = str(title)

self.\_\_track\_length = str(length)

except Exception as e:

raise Exception('Error setting initial values:\n' + str(e))

# -- Properties -- #

@property

def position(self):

return self.\_\_track\_position

@position.setter

def position(self, value):

try:

self.\_\_track\_position = int(value)

except Exception:

raise Exception('position needs to be Integer')

@property

def title(self):

return self.\_\_track\_title

@title.setter

def title(self, value):

try:

self.\_\_track\_title = str(value)

except Exception:

raise Exception('title needs to be String')

@property

def length(self):

return self.\_\_track\_length

@length.setter

def length(self, value):

try:

self.\_\_track\_length = str(value)

except Exception:

raise Exception('length needs to be String')

# -- Methods -- #

def \_\_str\_\_(self):

"""Returns Track details as formatted string"""

return '{}\t{}\t{}\n'.format(self.\_\_track\_position, self.\_\_track\_title, self.\_\_track\_length)

def get\_record(self) -> str:

"""Returns: Track record formatted for saving to file"""

return '{},{},{}\n'.format(self.position, self.title, self.length)

FileIO

class FileIO:

"""Processes data to and from file:

properties:

methods:

save\_inventory(file\_name, lst\_Inventory): -> None

load\_inventory(file\_name): -> (a list of CD objects)

"""

@staticmethod

def save\_inventory(file\_name: list, lst\_Inventory: list) -> None:

"""

Args:

file\_name (list): list of file names [CD Inventory, Track Inventory] that hold the data.

lst\_Inventory (list): list of CD objects.

Returns:

None.

"""

try:

trackfile = open(file\_name[1], 'w')

with open(file\_name[0], 'w') as file:

for disc in lst\_Inventory:

file.write(disc.get\_record())

for track in disc.cd\_tracks:

trackfile.write(str(disc.cd\_id) + "," + track.get\_record())

trackfile.close()

except Exception as e:

print('There was a general error!', e, e.\_\_doc\_\_, type(e), sep='\n')

@staticmethod

def load\_inventory(file\_name: list) -> list:

"""

Args:

file\_name (list): list of file names [CD Inventory, Track Inventory] that hold the data.

Returns:

list: list of CD objects.

"""

lst\_Inventory = []

try:

with open(file\_name[0], 'r') as file:

for line in file:

data = line.strip().split(',')

row = DC.CD(data[0], data[1], data[2])

lst\_Inventory.append(row)

with open(file\_name[1], 'r') as file:

for line in file:

data = line.strip().split(',')

PC.DataProcessor.select\_cd(lst\_Inventory, int(data[0])).add\_track(DC.Track(int(data[1]),data[2],data[3]))

except Exception as e:

print('There was a general error!', e, e.\_\_doc\_\_, type(e), sep='\n')

return lst\_Inventory