```
"""
Ryan Kennedy, Gabriel Walder
Cmdr. Schenk
Cloud Computing
7th Period
March 18, 2025
"""

import tkinter as tk
from tkinter import ttk

from database import Database
from author_record import AuthorRecord
from book_record import BookRecord

class GUI():

    def __init__(self):

        self.root = tk.Tk()
        self.root.geometry("1000x1000")

        self.db = Database()

        self.authors_records = self.db.authors_get_all_records()
        self.authors_selected_record_index = 0

        self.books_records = self.db.books_get_all_records()
        self.books_selected_record_index = 0

        self.canvas = tk.Canvas(self.root, width=1000, height=1000)
        self.canvas.pack()

        self.feedback_lbl = ttk.Label(self.root, text="", font=("Helvetica", 14))
        self.feedback_lbl.place(x=500, y=10, anchor="center")

        self.authors_create_gui_elements()
        self.books_create_gui_elements()

        self.refresh_display()

        self.root.mainloop()


    def authors_create_gui_elements(self):

        self.authors_display_lbl = ttk.Label(self.root, text="Authors:", font=("Helvet
ica", 18))
        self.authors_display_lbl.place(x=500,y=35)
        self.canvas.create_rectangle(300, 75, 800, 375, outline="black", width=3)
        # record display
        self.authors_id_lbl = ttk.Label(self.root, text="ID:", font=("Helvetica", 14))
        self.authors_id_dpy = ttk.Label(self.root, text="", font=("Helvetica", 14))
        self.authors_id_lbl.place(x=475,y=130)
        self.authors_id_dpy.place(x=525,y=130)

        self.authors_name_lbl = ttk.Label(self.root, text="Name:", font=("Helvetica",
14))
        self.authors_name_ent = ttk.Entry(self.root, width=20 )
        self.authors_name_lbl.place(x=460, y=160)
        self.authors_name_ent.place(x=540, y=160)

        self.authors_birth_year_lbl = ttk.Label(self.root, text="Birth Year:")
        self.authors_birth_year_ent = ttk.Entry(self.root, width=20 )
        self.authors_birth_year_lbl.place(x=460, y=190)
        self.authors_birth_year_ent.place(x=540, y=190)


        # navigation buttons
```

```
        self.authors_back_all_btn = ttk.Button(self.root, text="|<", command=lambda: s
elf.authors_increment_selected_record_index(-self.authors_selected_record_index))
        self.authors_back_all_btn.place(x=340, y=220)

        self.authors_back_5_btn = ttk.Button(self.root, text="<<", command=lambda: sel
f.authors_increment_selected_record_index(-5))
        self.authors_back_5_btn.place(x=410, y=220)

        self.authors_back_1_btn = ttk.Button(self.root, text="<", command=lambda: self
.authors_increment_selected_record_index(-1))
        self.authors_back_1_btn.place(x=480, y=220)

        self.authors_next_1_btn = ttk.Button(self.root, text=">", command=lambda: self
.authors_increment_selected_record_index(1))
        self.authors_next_1_btn.place(x=550, y=220)

        self.authors_next_5_btn = ttk.Button(self.root, text=">>", command=lambda: sel
f.authors_increment_selected_record_index(5))
        self.authors_next_5_btn.place(x=620, y=220)

        self.authors_next_all_btn = ttk.Button(self.root, text=">|", command=lambda: s
elf.authors_increment_selected_record_index(len(self.authors_records) - self.authors_s
elected_record_index))
        self.authors_next_all_btn.place(x=690, y=220)

        # CRUD

        self.authors_insert_btn = ttk.Button(self.root, text="Insert", command=self.au
thors_insert_record)
        self.authors_insert_btn.place(x=430, y=250)

        self.authors_update_btn = ttk.Button(self.root, text="Update", command=self.au
thors_update_current_record)
        self.authors_update_btn.place(x=500, y=250)

        self.authors_delete_btn = ttk.Button(self.root, text="Delete", command=self.au
thors_delete_current_record)
        self.authors_delete_btn.place(x=570,y=250)


    def authors_increment_selected_record_index(self, amt):
        self.authors_selected_record_index += amt

        if (self.authors_selected_record_index >= len(self.authors_records)):
            self.authors_selected_record_index = len(self.authors_records) - 1

        if (self.authors_selected_record_index < 0):
            self.authors_selected_record_index = 0

        self.refresh_display()

    def authors_display_record_at_index(self, index):
        if (len(self.authors_records) == 0):
            blank = AuthorRecord()
            blank.fill(-1, "No Records", 0)
            self.authors_display_record(blank)
            return

        if (index < len(self.authors_records) and index >= 0):
            self.authors_display_record(self.authors_records[index])
        else:
            self.feedback_lbl["text"] = "ERROR: Tried to display index out of bounds."

    def authors_display_record(self, record):
        self.authors_id_dpy["text"] = str(record.id)
        self.authors_name_ent.delete(0, "end")
        self.authors_name_ent.insert(0, record.name)
        self.authors_birth_year_ent.delete(0, "end")
```

```
        self.authors_birth_year_ent.insert(0, str(record.birth_year))

    def authors_insert_record(self):
        if (self.authors_validate_input() == True):
            rec = AuthorRecord()
            rec.fill(-1, self.authors_name_ent.get(), int(self.authors_birth_year_ent.
get()))
            self.db.authors_insert(rec)
            self.authors_records = self.db.authors_get_all_records()
            self.refresh_display()

    def authors_delete_current_record(self):
        current_record = self.authors_records[self.authors_selected_record_index]
        self.db.authors_delete(current_record.id)
        self.authors_records = self.db.authors_get_all_records()
        self.books_records = self.db.books_get_all_records()
        self.refresh_display()

    def authors_update_current_record(self):
        if (self.authors_validate_input() == True):
            rec = AuthorRecord()
            rec.fill(self.authors_records[self.authors_selected_record_index].id, self
.authors_name_ent.get(), int(self.authors_birth_year_ent.get()))
            self.db.authors_update(rec)
            self.authors_records = self.db.authors_get_all_records()
            self.refresh_display()

    def authors_validate_input(self):
        name_text = self.authors_name_ent.get()
        birth_year_text = self.authors_birth_year_ent.get()

        if (self.is_character_in_string(name_text, '\'') or self.is_character_in_strin
g(name_text, ';') or self.is_character_in_string(birth_year_text, '\'') or self.is_cha
racter_in_string(birth_year_text, ';')):
            self.feedback_lbl["text"] = "You cannot have ' or ; in your inputs."
            return False

        try:
            birth_year_num = int(birth_year_text)
        except:
            self.feedback_lbl["text"] = "Birth year has to be a valid number."
            return False

        return True

    # ==== Books Stuff ====

    def books_create_gui_elements(self):

        self.books_display_lbl = ttk.Label(self.root, text="Books:", font=("Helvetica"
, 18))
        self.books_display_lbl.place(x=500,y=540)
        self.canvas.create_rectangle(300, 575, 800, 875, outline="black", width=3)
        # record display
        self.books_id_lbl = ttk.Label(self.root, text="ID:", font=("Helvetica", 14))
        self.books_id_dpy = ttk.Label(self.root, text="", font=("Helvetica", 14))
        self.books_id_lbl.place(x=475,y=590)
        self.books_id_dpy.place(x=525,y=590)

        self.books_name_lbl = ttk.Label(self.root, text="Name:", font=("Helvetica", 14
))
        self.books_name_ent = ttk.Entry(self.root, width=20 )
        self.books_name_lbl.place(x=460, y=620)
        self.books_name_ent.place(x=540, y=620)

        self.books_year_released_lbl = ttk.Label(self.root, text="Published Year:", fo
nt=("Helvetica", 14))
        self.books_year_released_ent = ttk.Entry(self.root, width=20 )
        self.books_year_released_lbl.place(x=430, y=650)
```

```
        self.books_year_released_ent.place(x=580, y=650)

        self.books_page_amt_lbl = ttk.Label(self.root, text="# of Pages:", font=("Helv
etica", 14))
        self.books_page_amt_ent = ttk.Entry(self.root, width=20 )
        self.books_page_amt_lbl.place(x=430, y=680)
        self.books_page_amt_ent.place(x=580, y=680)

        self.books_price_lbl = ttk.Label(self.root, text="Price:", font=("Helvetica",
14))
        self.books_price_ent = ttk.Entry(self.root, width=20 )
        self.books_price_lbl.place(x=460, y=710)
        self.books_price_ent.place(x=540, y=710)

        self.books_author_lbl = ttk.Label(self.root, text="Author:", font=("Helvetica"
, 14))
        self.books_author_val = tk.StringVar()
        self.books_author_cbx = ttk.Combobox(self.root, textvariable=self.books_author
_val)
        self.books_author_cbx["values"] = [""]
        self.books_author_cbx["state"] = "readonly"
        self.books_author_val.set("")
        self.books_author_lbl.place(x=460, y=740)
        self.books_author_cbx.place(x=540, y=740)


        # navigation buttons

        self.books_back_all_btn = ttk.Button(self.root, text="|<", command=lambda: sel
f.books_increment_selected_record_index(-self.books_selected_record_index))
        self.books_back_all_btn.place(x=340, y=770)

        self.books_back_5_btn = ttk.Button(self.root, text="<<", command=lambda: self.
books_increment_selected_record_index(-5))
        self.books_back_5_btn.place(x=410, y=770)

        self.books_back_1_btn = ttk.Button(self.root, text="<", command=lambda: self.b
ooks_increment_selected_record_index(-1))
        self.books_back_1_btn.place(x=480, y=770)

        self.books_next_1_btn = ttk.Button(self.root, text=">", command=lambda: self.b
ooks_increment_selected_record_index(1))
        self.books_next_1_btn.place(x=550, y=770)

        self.books_next_5_btn = ttk.Button(self.root, text=">>", command=lambda: self.
books_increment_selected_record_index(5))
        self.books_next_5_btn.place(x=620, y=770)

        self.books_next_all_btn = ttk.Button(self.root, text=">|", command=lambda: sel
f.books_increment_selected_record_index(len(self.books_records) - self.books_selected_
record_index))
        self.books_next_all_btn.place(x=690, y=770)

        # CRUD

        self.books_insert_btn = ttk.Button(self.root, text="Insert", command=self.book
s_insert_record)
        self.books_insert_btn.place(x=430, y=800)

        self.books_update_btn = ttk.Button(self.root, text="Update", command=self.book
s_update_current_record)
        self.books_update_btn.place(x=500, y=800)

        self.books_delete_btn = ttk.Button(self.root, text="Delete", command=self.book
s_delete_current_record)
        self.books_delete_btn.place(x=570, y=800)


    def books_increment_selected_record_index(self, amt):
```

```
        self.books_selected_record_index += amt

        if (self.books_selected_record_index >= len(self.books_records)):
            self.books_selected_record_index = len(self.books_records) - 1

        if (self.books_selected_record_index < 0):
            self.books_selected_record_index = 0

        self.refresh_display()

    def books_display_record_at_index(self, index):
        if (len(self.books_records) == 0):
            blank = BookRecord()
            blank.fill(-1, "No Records", 0, 0, 0, -1)
            self.books_display_record(blank, "No Records")
            return

        if (index < len(self.books_records) and index >= 0):
            self.books_display_record(self.books_records[index], self.books_format_cbx
_value(self.books_records[index].author_id))
        else:
            self.feedback_lbl["text"] = "ERROR: Tried to display index out of bounds."

    def books_display_record(self, record, cbx_value):
        self.books_id_dpy["text"] = str(record.id)
        self.books_name_ent.delete(0, "end")
        self.books_name_ent.insert(0, record.name)
        self.books_year_released_ent.delete(0, "end")
        self.books_year_released_ent.insert(0, str(record.year_released))
        self.books_page_amt_ent.delete(0, "end")
        self.books_page_amt_ent.insert(0, str(record.page_amt))
        self.books_price_ent.delete(0, "end")
        self.books_price_ent.insert(0, str(record.price))
        self.books_author_val.set(cbx_value)

    def books_format_cbx_value(self, id):
        for author in self.authors_records:
            if (author.id == id):
                return "{} - {}".format(str(id), author.name)

        return "{} is not a valid id".format(str(id))

    def books_get_author_id_from_cbx_value(self, text):
        arr = text.split(" ")
        return int(arr[0])

    def books_insert_record(self):
        if (self.books_validate_input() == True):
            rec = BookRecord()
            rec.fill(-1, self.books_name_ent.get(), int(self.books_year_released_ent.g
et()), int(self.books_page_amt_ent.get()), float(self.books_price_ent.get()), self.boo
ks_get_author_id_from_cbx_value(self.books_author_val.get()))
            self.db.books_insert(rec)
            self.books_records = self.db.books_get_all_records()
            self.refresh_display()

    def books_delete_current_record(self):
        current_record = self.books_records[self.books_selected_record_index]
        self.db.books_delete(current_record.id)
        self.books_records = self.db.books_get_all_records()
        self.refresh_display()

    def books_update_current_record(self):
        if (self.books_validate_input() == True):
            rec = BookRecord()
            rec.fill(self.books_records[self.books_selected_record_index].id, self.boo
ks_name_ent.get(), int(self.books_year_released_ent.get()), int(self.books_page_amt_en
t.get()), float(self.books_price_ent.get()), self.books_get_author_id_from_cbx_value(s
elf.books_author_val.get()))
```

```
            self.db.books_update(rec)
            self.books_records = self.db.books_get_all_records()
            self.refresh_display()

    def books_validate_input(self):
        name_text = self.books_name_ent.get()
        year_released_text = self.books_year_released_ent.get()
        page_amt_text = self.books_page_amt_ent.get()
        price_text = self.books_price_ent.get()

        if (self.is_character_in_string(name_text, '\'') or self.is_character_in_strin
g(name_text, ';') or
            self.is_character_in_string(year_released_text, '\'') or self.is_character
_in_string(year_released_text, ';') or
            self.is_character_in_string(page_amt_text, '\'') or self.is_character_in_s
tring(page_amt_text, ';') or
            self.is_character_in_string(price_text, '\'') or self.is_character_in_stri
ng(price_text, ';')):

            self.feedback_lbl["text"] = "You cannot have ' or ; in your inputs."
            return False

        try:
            year_released = int(year_released_text)
            page_amt = int(page_amt_text)
            price = float(price_text)
        except:
            self.feedback_lbl["text"] = "Your inputs (year released, page amount, pric
e) aren't valid numbers."
            return False

        return True



    def refresh_display(self):

        if (self.authors_selected_record_index >= len(self.authors_records)):
            self.authors_selected_record_index = len(self.authors_records) - 1

        if (self.authors_selected_record_index < 0):
            self.authors_selected_record_index = 0

        if (self.books_selected_record_index >= len(self.books_records)):
            self.books_selected_record_index = len(self.books_records) - 1

        if (self.books_selected_record_index < 0):
            self.books_selected_record_index = 0

        author_arr = []
        for author in self.authors_records:
            author_arr.append("{} - {}".format(author.id, author.name))

        self.books_author_cbx["values"] = author_arr

        self.authors_display_record_at_index(self.authors_selected_record_index)
        self.books_display_record_at_index(self.books_selected_record_index)

    def is_character_in_string(self, string, character):
        for c in string:
            if (c == character):
                return True
        return False
```

```
"""
Ryan Kennedy, Gabriel Walder
Cmdr. Schenk
Cloud Computing
7th Period
March 18, 2025
"""

import mysql.connector
import random

from author_record import AuthorRecord
from book_record import BookRecord

# CREATE DATABASE RyanKennedyAndGabrielWaldner;
# USE RyanKennedyAndGabrielWaldner;
# CREATE TABLE authors (
#     id integer auto_increment unique not null,
#     name text not null,
#     birth_year integer,
#     PRIMARY KEY (id)
# );
# CREATE TABLE books (
#     id integer auto_increment unique not null,
#     name text not null,
#     year_released integer,
#     page_amt integer,
#     price float,
#     author_id integer not null,
#     PRIMARY KEY (id),
#     FOREIGN KEY (author_id) REFERENCES authors(id)
# );

class Database():

    def __init__(self):
        # self.conn = mysql.connector.connect(host = "192.168.0.100", user = "student"
, passwd = "jchs", database = "RyanKennedyAndGabrielWaldner")
        self.conn = mysql.connector.connect(host = "127.0.0.1", user = "root", passwd
= "ryansmiles", database = "RyanKennedyAndGabrielWaldner")
        self.cursor = self.conn.cursor()

    def __del__(self):
        self.conn.commit()
        self.conn.close()

    def books_get_all_records(self):
        self.cursor.execute("SELECT * FROM books;")

        arr_data = self.cursor.fetchall()

        result = []

        if(len(arr_data) == 0):
            return result

        for record in arr_data:
            rec = BookRecord()
            rec.fill(record[0], record[1], record[2], record[3], record[4], record[5])
            result.append(rec)

        return result


    def authors_get_all_records(self):
        self.cursor.execute("SELECT * FROM authors;");

        arr_data = self.cursor.fetchall()
```

```
        result = []

        if(len(arr_data) == 0):
            return result

        for record in arr_data:
            rec = AuthorRecord()
            rec.fill(record[0], record[1], record[2])
            result.append(rec)


        return result


    def books_insert(self, record):
        self.cursor.execute("INSERT INTO books (name, year_released, page_amt, price,
author_id) VALUES ('{}', {}, {}, {}, {});".format(record.name, record.year_released, r
ecord.page_amt, record.price, record.author_id))

    def authors_insert(self, record):
        self.cursor.execute("INSERT INTO authors (name, birth_year) VALUES ('{}', {});
".format(record.name, record.birth_year))

    def books_update(self, record):
        self.cursor.execute("UPDATE books SET name = '{}', year_released = {}, page_am
t = {}, price = {}, author_id ={} WHERE id = {};".format(record.name, record.year_rele
ased, record.page_amt, record.price, record.author_id, record.id))

    def authors_update(self, record):
        self.cursor.execute("UPDATE authors SET name = '{}', birth_year = {} WHERE id
= {};".format(record.name, record.birth_year, record.id))

    def books_delete(self, id):
        self.cursor.execute("DELETE FROM books WHERE id = {};".format(id))

    def authors_delete(self, id):
        self.cursor.execute("DELETE FROM books WHERE author_id = {};".format(id))
        self.cursor.execute("DELETE FROM authors WHERE id = {};".format(id))
```

```
"""
Ryan Kennedy, Gabriel Walder
Cmdr. Schenk
Cloud Computing
7th Period
March 18, 2025
"""

# CREATE TABLE authors (
#     id integer auto_increment unique not null,
#     name text not null,
#     birth_year integer,
#     PRIMARY KEY (id)
# );

class AuthorRecord:

    def __init__(self):
        self.id = -1
        self.name = "None"
        self.birth_year = 0

    def fill(self, id, name, birth_year):
        self.id = id
        self.name = name
        self.birth_year = birth_year
```

```
"""
Ryan Kennedy, Gabriel Walder
Cmdr. Schenk
Cloud Computing
7th Period
March 18, 2025
"""

# CREATE TABLE books (
#     id integer auto_increment unique not null,
#     name text not null,
#     year_released integer,
#     page_amt integer,
#     price float,
#     author_id integer not null,
#     PRIMARY KEY (id),
#     FOREIGN KEY (author_id) REFERENCES authors(id)
# );

class BookRecord:

    def __init__(self):
        self.id = -1
        self.name = "None"
        self.year_released = 0
        self.page_amt = 0
        self.price = 0
        self.author_id = 0

    def fill(self, id, name, year_released, page_amt, price, author_id):
        self.id = id
        self.name = name
        self.year_released = year_released
        self.page_amt = page_amt
        self.price = price
        self.author_id = author_id
```

```
"""
Ryan Kennedy, Gabriel Walder
Cmdr. Schenk
Cloud Computing
7th Period
March 18, 2025
"""

import sys
from gui import GUI

def main():
    gui = GUI()
    sys.exit(1)

if (__name__=="__main__"):
    main()
```

(kenne@metal

Authors:

ID: 11

Name: JK Rowling

Birth Year: 1970

|< << < > >> >|

Insert Update Delete

Books:

ID: 2

Name: Harry Potter

Published Year: 1990

# of Pages: 200

Price: 10.0

Author: 11 - JK Rowling

|< << < > >> >|

Insert Update Delete