

تصميم نظام OOP لعقارات

❖ الفئات الرئيسية

1. ** العقار (Property)

- خصائص: عنوان، مساحة، غرف، حمامات، سعر الشراء، سعر الإيجار.

- أوامر: عرض التفاصيل، تحديث السعر.

2. ** المالك (Owner)

- خصائص: اسم، عنوان، هاتف، بريد إلكتروني.

- أوامر: عرض الملف الشخصي، إضافة عقار، حذف عقار.

3. ** الوكيل (Agent)

- خصائص: اسم، عنوان، هاتف، بريد إلكتروني، لجنة.

- أوامر: عرض الملف الشخصي، إضافة عقار، حذف عقار.

4. ** العميل (Client)

- خصائص: اسم، عنوان، هاتف، بريد إلكتروني.

- أوامر: عرض الملف الشخصي، البحث عن عقار، تقديم طلب شراء/إيجار.

❖ الفئات الفرعية

1. ** عقار للبيع (SaleProperty): يرث من العقار.

- خصائص إضافية: سعر البيع، حالة العقار.

- أوامر إضافية: تحديث حالة البيع.

2. ** عقار للإيجار (RentalProperty) : يوث من العقار.

- خصائص إضافية: سعر الإيجار، مدة الإيجار.

- أوامر إضافية: تحديد حالة الإيجار.

❖ الأوامر

1. بيع العقار (SellProperty) : يؤديه المالك أو الوكيل.

- معالجة: تحديد حالة البيع، إضافة العميل، تحديد سعر البيع.

2. شراء العقار (BuyProperty) : يؤديه العميل.

- معالجة: تحديد حالة البيع، إضافة العميل، دفع الثمن.

3. إيجار العقار (RentProperty) : يؤديه العميل.

- معالجة: تحديد حالة الإيجار، إضافة العميل، دفع الإيجار.

❖ العلاقات

1. المالك يملك عقارات متعددة.

2. الوكيل يدير عقارات متعددة.

3. العميل يمكنه شراء أو إيجار عقار واحد أو أكثر.

4. العقار يمكن أن يكون للبيع أو للإيجار.

class Property:

```
def __init__(self, title, area, rooms, bathrooms, price):  
    self.title = title  
    self.area = area  
    self.rooms = rooms  
    self.bathrooms = bathrooms  
    self.price = price  
  
    def display_details(self):  
        print(f'{self.title} , المساحة: {self.area} م², الغرف: {self.rooms}, الحمامات: {self.bathrooms}, السعر: {self.price}')"
```

class Owner:

```
def __init__(self, name, address, phone, email):  
    self.name = name  
    self.address = address  
    self.phone = phone  
    self.email = email  
  
    self.properties[] =
```

```
def add_property(self, property):  
    self.properties.append(property)
```

class Agent:

```
def __init__(self, name, address, phone, email, commission):  
    self.name = name  
    self.address = address  
    self.phone = phone  
    self.email = email  
    self.commission = commission  
    self.properties[] =
```

```
def add_property(self, property):  
    self.properties.append(property)
```

```
class Client:  
    def __init__(self, name, address, phone, email):  
        self.name = name  
        self.address = address  
        self.phone = phone  
        self.email = email  
    def buy_property(self, property):  
        print(f'{self.name} من قبل تم شراء العقار {property.title}')
```

```
def rent_property(self, property):  
    print(f'{self.name} من قبل تم إيجار العقار {property.title}')
```

❖ إنشاء عقار

(500000 ,2 ,3 ,150)property1 = Property

❖ إنشاء مالك

("mohamed@example.com" , "0123456789" , "القاهرة" , "محمد")owner1 = Owner

owner1.add_property(property1)

❖ إنشاء عميل

("ahmed@example.com" , "0123456789" , "القاهرة" , "أحمد")client1 = Client

client1.buy_property(property1)

تصميم نظام إجرائي (Procedural) لجامعة:

❖ الهيكل العام

1. مستودع البيانات (Data Storage): يحتوي على معلومات الطلاب والأساتذة والدورات.
2. وظائف الإدارة (Management Functions): لإدارة الجامعة.
3. وظائف العمليات (Operation Functions): لتنفيذ العمليات اليومية.

❖ مستودع البيانات

1. جدول الطلاب (Students) (id -

name -

major -

GPA -

1. جدول الأساتذة (Professors)

id -

name -

department -

courses -

1. جدول الدورات (Courses)

id -

name -

credits -

professor_id -

1. جدول التسجيلات (Enrollments)

student_id -

course_id -

grade -

❖ وظائف الإدارة

1. إضافة طالب (add_student)

2. حذف طالب (delete_student)

3. إضافة أستاذ (add_professor)

4. حذف أستاذ (delete_professor)

5. إضافة دورة (add_course)

6. حذف دورة (delete_course)

7. تسجيل طالب في دورة (enroll_student)

8. حذف تسجيل طالب من دورة (drop_course)

❖ وظائف العمليات

1. عرض معلومات طالب (display_student_info)

2. عرض معلومات أستاذ (display_professor_info)

3. عرض معلومات دورة (display_course_info)

4. تحديث درجة طالب في دورة (update_grade)

5. حساب معدل طالب (calculate_GPA)

❖ الأمثلة بلغة البايثون

❖ مستودع البيانات

students[] =

professors[] =

courses[] =

enrollments[] =

❖ وظائف الإِدَارَة

```
def add_student(name, major):  
    students.append({'id': len(students) + 1, 'name': name, 'major': major, 'GPA':  
0.0})  
  
def add_professor(name, department):  
    professors.append({'id': len(professors) + 1, 'name': name, 'department':  
department, 'courses[] :'})  
  
def add_course(name, credits, professor_id):  
    courses.append({'id': len(courses) + 1, 'name': name, 'credits': credits,  
'professor_id': professor_id})  
  
def enroll_student(student_id, course_id):  
    enrollments.append({'student_id': student_id, 'course_id': course_id, 'grade':  
None})
```

❖ وظائف العمليات

```
def display_student_info(student_id):  
    for student in students:  
        if student['id'] == student_id:  
            print(f"معلومات الطالب: {student['name']}, {student['major']}, {student['GPA']}")  
    Return  
    print("الطالب غير موجود")  
  
def calculate_GPA(student_id):  
    grades = [enrollment['grade'] for enrollment in enrollments if  
enrollment['student_id'] == student_id]
```

GPA = sum(grades) / len(grades)

return GPA

❖ إنشاء طالب

("أحمد محمد", "هندسة")add_student

❖ إنشاء أستاذ

("د. علي خالد", "هندسة")add_professor

❖ إنشاء دورة

(1 ,3 ,)"برمجة")add_course

❖ تسجيل طالب في دورة

(1 ,1)enroll_student

❖ عرض معلومات طالب

(1)display_student_info

❖ حساب معدل طالب

(1)GPA = calculate_GPA

("{GPA} {معدل الطالب:} f)print