جزوه برنامه نویسی پیشرفته

امیرحسین قلی زاده

دانشگاه علوم و فنون آریان ترم دوم ۱۴۰۳ – ۱۴۰۴

جزوه آموزشی – جلسه دوم: برنامه‌نویسی شیءگرا ۱ (OOP)

استاد: امیرحسین قلی زاده | **دانشگاه:** علوم و فنون آریان | **نیمسال تحصیلی:** نیمسال دوم ۱۴۰۳ - ۱۴۰۴

فهرست مطالب

جزوه آموزشی – جلسه دوم: برنامه‌نویسی شیءگرا 1 (OOP) ۱
مقدمه ۲
۱. کلاس‌ها و اشیاء ۲
۱.۱. تعریف کلاس ۲
۱.۲. تعریف شی (Object) ۲
۱.۳. مثال ساده در پایتون ۲
۲. سازنده‌ها (Constructors) و متدها (Methods) ۳
۲.۱. سازنده (Constructor) ۳
۲.۲. متدها (Methods) ۳
مثال از سازنده و متدها ۳
۳. وراثت (Inheritance) ۴
۳.۱. مفهوم وراثت ۴
۳.۲. مثال از وراثت ۴
۴. چندریختی (Polymorphism) ۵
۴.۱. تعریف چندریختی ۵
۴.۲. مثال از چندریختی ۵
۵. تمرین‌های کاربردی ۵
۵.۲. تکلیف گروهی: کلاس Library ۶
نتیجه‌گیری ۶
منابع تکمیلی برای مطالعه بیشتر ۷

مقدمه

برنامه‌نویسی شیءگرا (OOP) یکی از مهم‌ترین پارادایم‌های برنامه‌نویسی است که به توسعه‌دهندگان امکان می‌دهد تا با استفاده از کلاس‌ها و اشیاء، کدهایی منظم‌تر، قابل نگهداری‌تر و قابل استفاده مجدد بنویسند. در این جلسه، به مفاهیم پایه‌ای مانند کلاس‌ها، اشیاء، سازنده‌ها، متدها، وراثت و چندریختی پرداخته می‌شود.

---

۱. کلاس‌ها و اشیاء

۱.۱. تعریف کلاس

کلاس یک الگوی (Blueprint) کلی برای ایجاد اشیاء (نمونه‌ها) است. در واقع، کلاس‌ها مانند «طرح»هایی هستند که مشخص می‌کنند اشیاء باید چه ویژگی‌ها (attributes) و متدهایی (methods) داشته باشند.

۱.۲. تعریف شی (Object):

شیء نمونه‌ای از یک کلاس است. وقتی که از یک کلاس، یک شیء می‌سازید، در واقع یک موجودیت واقعی با ویژگی‌های مشخص شده توسط کلاس به دست می‌آورید. (مانند ساختن آقای/خانم X از روی کلاس انسان!)

۱.۳. مثال ساده در پایتون

```python
class Person:
    def __init__(self, name, age):
        self.name = name      # ویژگی نام
        self.age = age        # ویژگی سن

    def introduce(self):
        print(f"سلام، من {self.name} و هستم {self.age} سال دارم.")

# ایجاد یک شیء از کلاس Person
person1 = Person("علی", 25)
person1.introduce()  # خروجی: سلام، من علی هستم و 25 سال دارم.

```

در این مثال، کلاس Person شامل یک سازنده (__init__) و یک متد به نام introduce است. سازنده برای مقداردهی اولیه به ویژگی‌های شیء استفاده می‌شود.

---

۲. سازنده‌ها (Constructors) و متدها (Methods)

۲.۱. سازنده (Constructor)

سازنده متدی است که بلافاصله پس از ایجاد شیء اجرا می‌شود تا ویژگی‌های اولیه را تنظیم کند. در پایتون از متد `__init__` استفاده می‌شود.

- ویژگی‌ها: مقداردهی اولیه، تعیین وضعیت اولیه شیء.

۲.۲. متدها (Methods)

متدها توابعی (Function) هستند که در داخل کلاس تعریف می‌شوند و می‌توانند بر روی ویژگی‌های شیء عمل کنند.

مثال: متدهایی برای نمایش اطلاعات، تغییر ویژگی‌ها و انجام عملیات مختلف.

مثال از سازنده و متدها:

```python
class Car:
    def __init__(self, model, year, color):
        self.model = model    # مدل خودرو
        self.year = year      # سال ساخت
        self.color = color    # رنگ خودرو

    def display_info(self): # متد display_info
        print(f"مدل: {self.model}, سال ساخت: {self.year}, رنگ: {self.color}")

    def change_color(self, new_color):
        self.color = new_color
        print(f" رنگ خودرو به {self.color} تغییر کرد.")

# استفاده از کلاس Car
my_car = Car("Toyota Corolla", 2020, "قرمز")
my_car.display_info()
my_car.change_color("آبی")
my_car.display_info()

```

در این مثال، کلاس Car دارای سازنده‌ای برای تنظیم ویژگی‌های اولیه و دو متد برای نمایش اطلاعات و تغییر رنگ خودرو است.

---

۳. وراثت (Inheritance)

۳.۱. مفهوم وراثت

وراثت اجازه می‌دهد تا یک کلاس (زیرکلاس) از ویژگی‌ها و رفتارهای یک کلاس دیگر (کلاس والد / Parent Class) استفاده کند. این امر باعث کاهش کد تکراری و افزایش قابلیت استفاده مجدد می‌شود.

۳.۲. مثال از وراثت:

```
class Vehicle:
    def __init__(self, brand, year):
        self.brand = brand
        self.year = year

    def honk(self):
        print("بیپ بیپ!")

class Car(Vehicle):
    def __init__(self, brand, year, model, color):
        super().__init__(brand, year)   # فراخوانی سازنده کلاس پدر
        self.model = model
        self.color = color

    def display_info(self):
        print(f"برند: {self.brand}, مدل: {self.model}, سال: {self.year}, رنگ: {self.color}")

# استفاده از کلاس Car که از Vehicle برد وراثت می
my_car = Car("Honda", 2019, "Civic", "مشکی")
my_car.display_info()
my_car.honk()  # استفاده از متد وراثتی

```

در این مثال، کلاس Car از کلاس Vehicle ارث می‌برد و می‌تواند از متدهای آن (مثل honk) استفاده کند.

---

۴. چندریختی (Polymorphism)

۴.۱. تعریف چندریختی:

چندریختی به این معناست که اشیاء از کلاس‌های مختلف می‌توانند به‌عنوان نمونه‌هایی از یک کلاس پدر استفاده شوند و هر کدام رفتار مخصوص به خود را داشته باشند. این امر باعث می‌شود تا با یک رابط یکسان بتوان با انواع مختلف اشیاء کار کرد.

۴.۲. مثال از چندریختی:

```
class Animal:
    def make_sound(self):
        pass  # متدی بدون پیاده سازی

class Dog(Animal):
    def make_sound(self):
        print("واق واق!")

class Cat(Animal):
    def make_sound(self):
        print("میو میو!")

# استفاده از چندریختی
animals = [Dog(), Cat()]
for animal in animals:
    animal.make_sound()

```

در این مثال، با وجود اینکه هر دو کلاس Dog و Cat از کلاس Animal ارث می‌برند، هر کدام پیاده‌سازی خاص خود را برای متد make_sound دارند. این امکان را می‌دهد که به صورت یکسان از اشیاء مختلف استفاده کنیم.

---

۵. تمرین‌های کاربردی:

۵.۱. تکلیف شخصی: کلاس Car

وظیفه:

یک کلاس ساده به نام Car ایجاد کنید که ویژگی‌هایی مانند مدل، سال ساخت و رنگ را ذخیره کند و متدهایی برای نمایش اطلاعات خودرو و تغییر رنگ داشته باشد.

راهنمایی:

- در سازنده (__init__) اطلاعات خودرو را مقداردهی اولیه کنید.
- متد display_info برای نمایش اطلاعات خودرو تعریف کنید.
- متد change_color برای تغییر رنگ خودرو پیاده‌سازی کنید.

۵.۲. تکلیف گروهی: کلاس Library

وظیفه:

یک کلاس Library طراحی کنید که شامل اطلاعات کتاب‌ها (نام، نویسنده، تاریخ انتشار) و متدهایی برای افزودن و جستجو در کتاب‌ها باشد.

راهنمایی:

- از یک لیست یا دیکشنری برای نگهداری اطلاعات کتاب‌ها استفاده کنید.
- متد add_book برای افزودن کتاب جدید.
- متد search_book برای جستجو بر اساس نام یا نویسنده.

نتیجه‌گیری

در این جلسه با مفاهیم اصلی برنامه‌نویسی شیءگرا آشنا شدید:

- **کلاس‌ها و اشیاء:** نحوه تعریف الگوهای کلی و ایجاد نمونه‌های واقعی.
- **سازنده‌ها و متدها:** نحوه مقداردهی اولیه به اشیاء و پیاده‌سازی رفتارهای آن‌ها.
- **وراثت:** استفاده از قابلیت به اشتراک گذاشتن ویژگی‌ها و متدها بین کلاس‌ها.
- **چندریختی:** توانایی استفاده از یک رابط یکسان برای اشیاء مختلف.

تمرین‌های شخصی و گروهی به شما کمک می‌کنند تا مفاهیم آموخته شده را به‌طور عملی به کار ببرید و درک عمیق‌تری از شیءگرایی پیدا کنید. توصیه می‌شود پس از مطالعه جزوه، نمونه‌های کد را خودتان اجرا کرده و با تغییرات مختلف، نقش هر بخش را تجزیه تحلیل کنید.

منابع تکمیلی برای مطالعه بیشتر:

- کتاب‌های مرجع شیءگرایی در پایتون یا زبان‌های دیگر مانند جاوا یا C#.
- دوره‌های آنلاین معتبر در سایت‌هایی مانند Coursera، Udemy، YouTube یا مکتب‌خونه.
- مستندات رسمی زبان مورد استفاده (مانند مستندات پایتون برای اطلاعات بیشتر درباره کلاس‌ها و شیءگرایی).
- مهم تر از همه: کد بزنید!

با مطالعه و تمرین مستمر، مباحث شیءگرایی به بخشی از تفکر برنامه‌نویسی شما تبدیل خواهد شد.

موفق باشید.