Introduction to Programming

Rania Baghernejad

Java – part 12

OOP - 1

مباحث

تعریف شئ (Object) در برنامه‌نویسی شی‌گرا

در برنامه‌نویسی شی‌گرا(Object-Oriented Programming) ، شئ به نمونه‌ای از یک کلاس (Class) گفته می‌شود. کلاس، یک ساختار تعریف‌شده توسط برنامه‌نویس است که مشخصات (ویژگی‌ها) و قابلیت‌ها (رفتارها) را توصیف می‌کند. اما برای استفاده‌ی عملی از این تعریف، باید از روی کلاس، یک شئ ساخته شود.

به عبارت دقیق‌تر، شئ موجودیتی در حافظه است که بر پایه‌ی کلاس ایجاد می‌گردد و می‌تواند داده نگه‌داری کند و رفتار خاصی از خود بروز دهد.

چرا از شئ استفاده می‌شود؟

- ساختارهای پیچیده را مدل‌سازی کنیم؛ مانند دانشجو، کتاب، حساب بانکی و...
- داده‌ها و رفتارها را در یک واحد منطقی تجمیع کنیم
- قابلیت استفاده‌ی مجدد (Reusability) و قابلیت توسعه (Extensibility) در برنامه ایجاد کنیم

تفاوت کلاس و شئ

شئ(Object)	کلاس(Class)
یک نمونه واقعی از آن تعریف است	تعریف کلی از یک نوع موجودیت است
در زمان اجرا در حافظه ایجاد می‌شود	در زمان کامپایل فقط طرح است
Book b1 = new Book(); یک کتاب واقعی با مقادیر خاص ایجاد می‌کند	مثلا: کلاس Book تعریف می‌کند یک کتاب چه ویژگی‌ها و رفتارهایی دارد

نحوه ایجاد یک شئ

برای ساختن یک شئ، از عملگر new استفاده می‌شود که نمونه‌ای از کلاس را در حافظه ایجاد کرده و سازنده (Constructor) آن را فراخوانی می‌کند:

```
Book myBook = new Book();
```

به طور خلاصه

- کلاس‌ها تعریف‌های عمومی هستند که صرفاً نوع داده‌ها و عملکردها را مشخص می‌کنند.
- برای استفاده از یک کلاس، باید از آن شئ ایجاد کنیم.
- شئ‌ها داده‌های خاص خود را دارند و مستقل از یکدیگر عمل می‌کنند، حتی اگر از یک کلاس ساخته شده باشند.

کپسوله‌سازی(Encapsulation)

کپسوله‌سازی به این معناست که داده‌ها (مثل ویژگی‌های یک شی) باید از دسترسی مستقیم خارج از کلاس خود محدود شوند. این کار به این دلیل انجام می‌شود که اطلاعات داخلی یک شی نباید به‌راحتی از بیرون تغییر کنند. به جای دسترسی مستقیم به متغیرها، از متدهای عمومی Getter و Setter برای خواندن و نوشتن مقادیر استفاده می‌شود.

چرا کپسوله‌سازی مهم است؟

- از تغییرات ناخواسته در داده‌ها جلوگیری می‌کند.
- قابلیت کنترل و بررسی داده‌ها از طریق متدها را فراهم می‌کند.
- کد را امن‌تر و قابل نگهداری‌تر می‌سازد.

سازنده‌ها(Constructors)

سازنده‌ها متدهایی هستند که برای مقداردهی اولیه به ویژگی‌های شی در هنگام ساخت آن استفاده می‌شوند. به‌طور معمول، سازنده‌ها همان نام کلاس را دارند و نیازی به نوع بازگشتی (مثل voidیا int) ندارند.

چرا سازنده‌ها مهم هستند؟

- از آن‌ها برای مقداردهی اولیه به ویژگی‌ها هنگام ایجاد شی استفاده می‌شود.
- به‌راحتی می‌توان یک شی را با ویژگی‌های اولیه مختلف ایجاد کرد.

استفاده از this

در داخل یک سازنده یا متد، کلمه کلیدی this به‌طور خاص به شی جاری اشاره دارد. یعنی اگر از this.title استفاده می‌کنیم، منظور متغیر title در شی جاری است که به آن دسترسی داریم. (که به این منظور متد set را برای آن تنظیم می‌کنیم.)

توضیحات:

- کلمه کلیدی this: در اینجا، this.title به ویژگی title شی جاری اشاره دارد. اگر this نبود، ممکن بود متغیر محلی با همان نام باعث اشتباه شود.
- بدون استفاده از this، کد به‌طور خودکار متغیرهای محلی را می‌گیرد و مشکلی در دسترسی به ویژگی‌های شی ایجاد می‌شود، بنابراین استفاده از this برای تمایز دادن بین ویژگی‌های شی و پارامترهای ورودی ضروری است.

مثال Book و BookTest:

در این مثال، کلاس Book ویژگی‌های مربوط به یک کتاب را نمایش می‌دهد و کپسوله‌سازی به کمک متدهای getter و setter انجام می‌شود. کد زیر را بررسی کنید:

```
// Class representing a Book
class Book {
                //ابتدا باید متغیرهای نمونه را مشخص نمود. زمانی که پرایوت باشد در بیرون از کلاس به طور مستقیم قابل دسترسی نیست.
    // Private variables (can't be accessed directly outside the class)
    private String title;
    private String author;
    private int year;

// متد زیر به متد سازنده یا متد constructor شناخته می‌شود که باید توجه داشته باشید که نام آن با نام کلاس یکی باشد. این متد برای مقدار دهی اولی به متغیرهای نمونه استفاده می‌شود. (در پرانتز آن همانند متدهای دیگر متغیرهای مورد نیاز را با ذکر نوع آن مشخص می‌کنیم.)
    // Constructor to initialize Book object
    public Book(String title, String author, int year) {
        this.title = title;
        this.author = author;
        this.year = year;
    }

//در ادامه باید به ترتیب برای هر متغیر نمونه متد تنظیم کنند و دریافت کننده نوشت. (اگر قرار است محاسباتی روی متغیرهای نمونه انجام شود بهتر است در متد get آن متغیر نمونه نوشته شود.)
    // Getter method to access the title
    public String getTitle() {
        return title;
    }

    // Setter method to set a new title
    public void setTitle(String title) {
        this.title = title;
    }
```

```
    // Getter method to access the author
    public String getAuthor() {
        return author;
    }

    // Setter method to set a new author
    public void setAuthor(String author) {
        this.author = author;
    }

    // Getter method to access the year
    public int getYear() {
        return year;
    }

    // Setter method to set a new year
    public void setYear(int year) {
        this.year = year;
    }

// متد زیر جهت پرینت نوشته می‌شود تا در تکرار زیاد از نوشتن کدهای اضافی جلوگیری شود و با فراخوانی آن در متد main و کلاس BookTest
پرینت گرفته شود.
    // Method to display the book's information
    public void displayInfo() {
        System.out.println("Book Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Year: " + year);
    }
}
```

```
// Testing class
public class BookTest {
    public static void main(String[] args) {
// در این مثال : Book نوع کلاس است و book1 نام شئ است. () new Book دستور ایجاد شئ است که سازنده‌ی کلاس را اجرا می‌کند. حتما
توجه داشته باشید که داخل پرانتز Book() متغیرهای منطبق با متغیرهای نمونه تعریف شده در پرانتز متد کانستراکتور باشد.)
        // Creating an object of Book
        Book book1 = new Book("The Great Gatsby", "F. Scott Fitzgerald", 1925);

        // Accessing the information using getter methods
        System.out.println("Book Title: " + book1.getTitle());
        System.out.println("Author: " + book1.getAuthor());
        System.out.println("Year: " + book1.getYear());

        // Modifying information using setter methods
```

```
        book1.setTitle("The Great Gatsby (Updated)");
        book1.setAuthor("F. Scott Fitzgerald (Updated)");

        // Displaying updated information
        System.out.println("\nUpdated Book Info:");
        book1.displayInfo();
    }
}
```

توضیحات:

۱. متغیرهای خصوصی(private) : ویژگی‌های title، author و year به‌صورت خصوصی تعریف شده‌اند. این متغیرها فقط در داخل کلاس Book قابل دسترسی هستند.

۲. متدهای getter و setter: برای دسترسی به این متغیرها از متدهای عمومی getو setاستفاده می‌شود. این متدها به ما این امکان را می‌دهند که داده‌ها را کنترل کنیم و از تغییرات ناخواسته جلوگیری کنیم.

۳. متد displayInfo: این متد به‌طور ساده اطلاعات کتاب را نمایش می‌دهد.

٤. تست در BookTest: در کلاس BookTest، یک شی از کلاس Book ساخته می‌شود و از طریق متدهای getter و setter به آن دسترسی پیدا می‌کنیم. تغییرات در کتاب انجام می‌دهیم و پس از آن اطلاعات به‌روز شده را نمایش می‌دهیم.

بدون استفاده از this چه اتفاقی می‌افتد؟

اگر در سازنده‌ها یا متدها از this استفاده نکنیم و نام پارامترها با ویژگی‌های کلاس یکی باشند، ممکن است دچار ابهام شویم. در این صورت، برنامه به پارامترهای ورودی ارجاع می‌دهد نه به ویژگی‌های شی.

مثال بدون استفاده از this:

```
class Book {
    private String title;
    private String author;

    // Constructor without 'this'
    public Book(String title, String author) {
        title = title;  // This doesn't work as expected
        author = author;  // Same issue here
    }

    public void displayInfo() {
        System.out.println("Title: " + title);  // Title will not be set
```

```
        System.out.println("Author: " + author);  // Author will not be set
    }
}

public class BookTest {
    public static void main(String[] args) {
        Book book1 = new Book("1984", "George Orwell");
        book1.displayInfo();  // Output will be null because title and author were
not set correctly
    }
}
```

توضیحات:

- در این مثال، متغیرهای title و author به درستی مقداردهی نمی‌شوند زیرا پارامترهای ورودی با ویژگی‌های شی هم‌نام هستند. در این صورت، استفاده از this برای تمایز دادن بین پارامترهای ورودی و ویژگی‌های شی ضروری است.

نتیجه‌گیری:

- کپسوله‌سازی و سازنده‌ها مفاهیمی اساسی در شی‌گرایی هستند. با استفاده از متدهای getter و setter می‌توانیم دسترسی به ویژگی‌های داخلی شی را محدود کنیم و از تغییرات ناخواسته جلوگیری کنیم.
- کلمه کلیدی this برای تمایز دادن بین پارامترهای ورودی و ویژگی‌های شی ضروری است. این کار از بروز اشتباهات جلوگیری می‌کند و کد را قابل خواندن‌تر می‌سازد.