به نام خدا

دستور کار کارگاه برنامه‌نویسی پیشرفته

جلسه دوم

مفاهیم کلاس و شئ

مقدمه

در این جلسه قرار است تا با مفاهیم کلاس و شئ، تفاوت این دو و نحوه برنامه نویسی شئ‌گرا با جاوا آشنا شویم. در برنامه‌نویسی شئ‌گرا، مفاهیم همانند زندگی واقعی هستند. یک شئ که می‌تواند هر چیزی باشد (مثلا یک دانشجو)، تعدادی ویژگی مخصوص به خود دارد (مثلا یک دانشجو، با ویژگی‌های نام، نام خانوادگی، شماره دانشجویی و موجودی حساب کاربری مشخص می‌شود). همچنین یک شئ می‌تواند عملیاتی انجام دهد (مثلا یک دانشجو می‌تواند در یک کارگاه ثبت‌نام کند، یا موجودی حساب خود را افزایش دهد). همان‌طور که می‌بینید، دانشجوهای مختلف ویژگی‌های مختلفی دارند و در آنِ واحد می‌توانند عملیات مختلفی مستقل از یکدیگر انجام دهند؛ ولی همه آنها در ویژگی‌ها و عملیاتی که می‌توانند انجام دهند، مشترکند. یک کلاس در برنامه‌نویسی شئ‌گرا، ویژگی‌ها و عملیات کلی مشترک بین شئ‌ها را تعریف می‌کند. برای اینکه بهتر متوجه این موضوع و تفاوت‌های آن شوید، تمرین‌های این جلسه را به دقت انجام دهید.

نکات آموزشی

در برنامه‌نویسی شئ‌گرا همانند برنامه‌نویسی ساخت‌یافته از مفاهیم متغیرها، تابع‌ها (که در اینجا متد[1] گفته می‌شوند)، دستورات شرطی و حلقه‌ها، انتساب‌ها و عملگرهای ریاضی و منطقی استفاده می‌شود. این مفاهیم را در درس مبانی برنامه‌نویسی یاد گرفته‌ایم و در دستور کار اول نیز یادآوری شده بود.

برای اینکه کدهایمان خوانا باشند، به نحوه تعریف کلاس‌ها، شئ‌ها، متدها، متغیرها، چینش متدها و متغیرها در یک کلاس و نحوه کامنت‌گذاری دقت کنید. تمام این موارد را با مثال خواهیم دید.

[1] Method

مراحل انجام کار

به صورت مرحله به مرحله برنامه‌ای می‌نویسیم که کارگاه برنامه‌نویسی پیشرفته را شبیه‌سازی کند. همان‌طور که می‌دانید یک کارگاه از تعدادی دانشجو تشکیل شده است.

انجام دهید: با استفاده از کدهای زیر کلاس دانشجو را تعریف کنید:

```
/**
 * The Student class represents a student in a student
administration system.
 * It holds the student details relevant in our context.
 *
 * @author Ehsan
 * @version 0.0
 */
public class Student {
    fields

    constructors

    methods
}
```

در تعریف هر کلاس دو قسمت وجود دارد:

۱) Header: اطلاعات مربوط به کلاسی که تعریف می‌کنید را به شکل کامنت در بیرون از کلاس و در ابتدای فایل قرار دهید. این اطلاعات به شما کمک می‌کند تا برنامه‌ای خوانا داشته باشید و توسعه‌دهندگان دیگر نیز با خواندن این توضیحات، دیدی کلی از کلاسی که نوشتید به دست بیاورند. برای کامنت‌گذاری مانند زبان C، از /* */ برای چند خط و // برای یک خط استفاده می‌کنیم. در صورتی که پیش از تعریف هر کلاس یا هر متد توضیحات مورد نظر را در بین /** و */ قرار دهید، اطلاعاتی را برای JavaDoc مربوط به یک کلاس تهیه کرده‌اید که می‌توانید توضیحات را در قالب یک فایل مرتب html مستندسازی کنید. در جلسه چهارم به طور مفصل درباره JavaDoc صحبت خواهیم کرد اما نکات ابتدایی آن را در این جلسه مطرح می‌کنیم. تعدادی از کلمات کلیدی که با @ شروع شده‌اند، برای تولیدکردن JavaDoc کاربرد دارد. مثلا عبارت @author Ehsan، در مستندسازی که به صورت خودکار تولید خواهد شد، نویسنده این کلاس را Ehsan ثبت می‌کند.

۲)تعریف کلاس: برای تعریف کلاس از کلمه کلیدی class استفاده می‌کنیم. کلمه سمت چپ آن یعنی public سطح دسترسی کلاس را مشخص می‌کند، فعلا کلاس‌هایمان را public تعریف می‌کنیم؛ به این معنا که همه کلاس‌های دیگر

قادر هستند از آن استفاده کنند. کلمه سمت راست، اسم کلاس است. اسامی کلاس را به شکل CamelCase[۲] با حرف اول بزرگ تعریف می‌کنیم. محدوده کلاس هم با {} مشخص می‌شود. همان طور که می‌بینید بعد از فیلدها، constructor و متدها به ترتیب تعریف می‌شوند.

۱. فیلدها: ویژگی‌های مشترک بین دانشجوها را در این‌جا تعریف می‌کنیم. مثل نام، نام خانوادگی، شماره دانشجویی و نمره. فیلدها را این طور تعریف می‌کنیم: اول سطح دسترسی آن را با private مشخص می‌کنیم، در این صورت این متغیر فقط در محدوده کلاس Student قابل دسترسی است. بعد از آن نوع (type) آن مثلا int و در نهایت اسم آن را مشخص می‌کنیم. نام را به شکل camelCase با حرف اول کوچک تعریف می‌کنیم. برای خوانایی بیشتر کد همه فیلدها را همراه با کامنت توضیحات آن فیلد تعریف می‌کنیم.

```
public class Student {

    // the student's first name
    private String firstName;

    // the student's last name
    private String lastName;

    // the student ID
    private String id;

    //the grade
    private int grade;

}
```

۲. Constructor: اولین رویه‌ای[۳] است که بعد از ساختن یک شئ فراخوانی می‌شود و فیلدهای لازم را مقداردهی می‌کند و عملیات تعیین‌شده لازم دیگر را انجام می‌دهد.

با استفاده از param@ در کامنت قبل از constructor، شرح تک تک پارامترهای ورودی به آن را مشخص می‌کنیم. constructor را با سطح دسترسی public و بدون نوع خروجی و همنام با نام کلاس تعریف می‌کنیم. در ابتدای تعریف هر

[۲] یعنی کلمات به هم چسبیده هستند و حرف اول در کلمات میانی بزرگ نوشته می‌شوند. مثلا goodStudent.

[۳] Procedure

شئ از جنس Student، نام، نام خانوادگی و شماره دانشجویی دریافت می‌شود و فیلد مربوطه به هرکدام مقداردهی می‌شود. همچنین در این مثال، مقدار نمره به طور پیش‌فرض برای هی شئ از این جنس، صفر قرار می‌گیرد.

```java
/**
 * Create a new student with a given name and ID number.
 *
 * @param fName first name of student
 * @param lname last name of student
 * @param sID student ID
 */
public Student(String fName, String lname, String sID){
        firstName = fName;
        lastName = lname;
        id = sID;
        grade = 0;
}
```

۳. متدها: عملیاتی که شئ‌های از جنس Student قادر به انجام آن هستند در این قسمت تعریف می‌شوند.

برای تعریف هر متد مانند تعریف تابع در C عمل می‌کنیم؛ با این تفاوت که در تعریف آن ابتدا سطح دسترسی آن متد را مشخص می‌کنیم. مثلا public. همچنین برای متدهایی که خروجی دارند با @return اطلاعات مربوط به خروجی متد را برای تولید مستندات ثبت می‌کنیم.

اگر نیاز است که از بیرون از کلاس به آن فیلدها دسترسی داشته باشیم، متدهای setter و getter را برای آنها تعریف می‌کنیم. مزیت این روش در آن است که اگر بخواهیم در مقدار دهی به یک فیلد محدودیتی ایجاد کنیم، می‌توانیم آن محدودیت را در متد set مربوط به آن پیاده‌سازی کنیم. مثلا می‌خواهیم شماره دانشجویی یک رشته ۷ رقمی باشد نه بیشتر یا کمتر. اگر فقط فیلد مربوط را public تعریف کنیم، یک شئ دیگر از بیرون می‌تواند به هر شکلی به آن مقداردهی کند که مورد نظر ما نیست!

انجام دهید: باتوجه به توضیحات قسمت قبل کلاس Student را کامل کنید.

```java
/**
 * get the first name of student
 * @return firstName field
 */
public String getFirstName() {
        return firstName;
}

/**
 * @param firstName set first name of a student
 */
public void setFirstName(String fName) {
        firstName = fName;
}

/**
 * Print the student's last name and ID number to the
output terminal.
 */
public void print() {
        System.out.println(lastName + ", student ID: "
                        + id + ", grade: " + grade);
}
```

سپس کد زیر را در یک کلاس جدید تعریف کنید. پروژه خود را اجرا کنید و خروجی خود را تحلیل کرده به مدرس کارگاه ارائه دهید.

```
public class Run {
    public static void main(String[] args) {
        Student std1 = new Student("Ehsan","Edalat", "9031066");
        Student std2 = new Student("Seyed", "Ahmadpanah", "9031806");
        Student std3 = new Student("Ahmad", "Asadi", "9031054");

        std1.print();
        std1.setGrade(15);
        std1.print();

        std2.print();
        std2.setGrade(11);
        std2.print();

        std3.print();
        std3.setFirstName("HamidReza");
        std3.print();
    }
}
```

همان‌طور که می‌بینید برای تعریف یک شئ (مانند خط سوم) این طور عمل می‌کنیم:

۱. ابتدا نوع شئ (نام کلاس) را تعیین می‌کنیم مثلا Student
۲. یک نام برای شئ انتخاب می‌کنیم. نام باید بامفهوم باشد و مرتبط با کلاسی باشد که از آن شئ را می‌سازیم. نام را به شکل camelCase با حرف اول کوچک می‌نویسیم.
۳. با استفاده از کلمه کلیدی **new** و به دنبال آن نام کلاس شئ ساخته می‌شود.
۴. درون پرانتز باید پارامترهایی که برای Constructor آن کلاس تعریف کرده‌ایم را مقداردهی کنیم.

انجام دهید: کلاس Lab را مانند کد زیر پیاده‌سازی کنید. کد زیر پیاده‌سازی متدها و constructor و کامنت‌های مرتبط با آنها را در خود ندارد. این موارد باید پیاده‌سازی شوند. در کلاس Run که پیش از این پیاده‌سازی کرده بودید، یک شئ از جنس Lab بسازید و تعدادی دانشجو به آن enroll کرده و در نهایت متد print را فراخوانی کنید. متد print باید شامل اطلاعات دانشجوهای Lab و میانگین نمره‌های آن‌ها باشد.

توجه! همان طور که می‌بینید در متد enrollStudent پارامتر ورودی از نوع Student است. یک متد از یک کلاس می‌تواند شئ ای از یک کلاس دیگر را به عنوان پارامتر ورودی دریافت کند.

فکر کنید: در مورد نحوه ارسال یک شئ به یک متد فکر کنید. این ارسال از جنس call-by-value است یا از جنس call-by-reference؟ با یک مثال درستی حدس خود را نشان دهید و به مدرس کارگاه نتیجه را گزارش دهید.

```
public class Lab {
    private Student[] students;
    private int avg;
    private String day;
    private int capacity;
    private int currentSize;
    public Lab(int cap, String d) {}

    public void enrollStudent(Student std) {
        if (currentSize < capacity) {
            students[currentSize] = std;
            currentSize++;
        } else {
            System.out.println("Lab is full!!!");
        }
    }
    public void print() {}
    public Student[] getStudents() {}
    public void setStudents(Student[] students) {}
    public int getAvg() {}
    public void calculateAvg() {}
    public String getDay() {}
    public void setDay(String day) {}
    public int getCapacity() {}
    public void setCapacity(int capacity) {}
}
```

اشکال‌زدایی

1 . دانشجویی متد print را به شکل زیر پیاده‌سازی کرده است. اشکال کد در کجاست؟

```
public void print() {
    for (int i = 0; i < students.size(); i++) {
        System.out.println("std fname: " + students[i].getFirstName()
                           + " std id:" + students[i].getId()
                           + " std grade:"+ students[i].getGrade());
    }
        System.out.println("Lab AVG:" + avg);
}
```

2 . با هماهنگی با مدرس کارگاه، کد پیاده‌سازی شده یکی دیگر از افراد کلاس را با توجه به نکات زیر ارزیابی کنید:

- کامنت‌گذاری مناسب و به‌جا برای فیلدها، متدها و constructorها و رعایت نکات مربوط به JavaDoc
- پیاده‌سازی درست کلاس‌ها و متدهای هر یک و آزمون درستی عملکرد آنها
- نحوه درست نام‌گذاری برای کلاس‌ها، متدها و فیلدها

انجام دهید:

می‌خواهیم شبیه سازی کلاس کارگاه را به دانشکده تعمیم دهیم. جزئیاتی که در مورد این شبیه‌سازی در دانشکده ما وجود دارد را ابتدا بر روی کاغذ بیاورید و در مورد آن با مدرس کارگاه مشورت کنید. سپس کلاس‌هایی که به آنها رسیده‌اید را پیاده‌سازی کنید. برای بررسی درستی کدهای خود کلاس Run مناسب پیاده‌سازی کنید. در انتها کد خود را به مدرس کارگاه ارائه دهید.