کارگاه برنامه نویسی پیشرفته

دستور کار شماره چهار

اهداف

آشنایی با کلاس‌های تغییرناپذیر[1]

آشنایی با کالکشن‌ها[2] در جاوا و انواع آن

[1] Immutable Class
[2] Collections

فهرست مطالب

کلاس‌های تغییرناپذیر ۳

تعریف کلاس تغییرناپذیر ۳

پیاده‌سازی یک کلاس تغییرناپذیر ۳

کالکشن‌ها در جاوا ۵

آشنایی با ArrayList ۵

آشنایی با لینکدلیست ۷

آشنایی با هش‌ست ۷

آشنایی با هش‌مپ ۹

آشنایی با Iteratorها ۱۱

معرفی Iterator ۱۱

انجام دهید ۱۲

کلاس Person ۱۲

کلاس Vote ۱۳

کلاس Voting ۱۵

کلاس VotingSystem ۱۸

کلاس Main ۲۰

تست برنامه ۲۰

کلاس‌های تغییرناپذیر

تعریف کلاس تغییرناپذیر

منظور از کلاس تغییرناپذیر این است که هنگامی که یک شئ از آن کلاس‌ها ایجاد می‌شود، نمی‌توانیم محتوای آن را تغییر دهیم. برای مثال تا اینجا با کلاس String آشنا شده‌اید. این کلاس یک کلاس تغییرناپذیر است، در نتیجه وقتی از آن یک شئ می‌سازیم، دیگر نمی‌توان محتوای آن را تغییر داد (با کلاس‌های تغییرناپذیر دیگر جاوا بعداً آشنا خواهید شد).

پیاده‌سازی یک کلاس تغییرناپذیر

حال ما نیز می‌توانیم کلاس‌های غیرقابل تغییری بسازیم. برای این کار مراحل زیر را انجام می‌دهیم:

۱. دسترسی فیلدهای کلاس را private تعریف می‌کنیم.
۲. با استفاده از کلیدواژه‌ی final، مقدار آن‌ها را ثابت می‌کنیم، به این معنا که زمانی که مقداردهی شدند دیگر قابل تغییر نیستند.
۳. درون کانستراکتور کلاس، همه‌ی فیلدها را مقداردهی می‌کنیم تا دیگر نیاز به متد دیگری برای مقداردهی نباشد.
۴. برای دسترسی به فیلدها برای آن‌ها گتر[1] می‌گذاریم.
۵. متدهای گتر فیلدهایی که نوع ابتدایی[2] نیستند را طوری پیاده‌سازی می‌کنیم که به جای برگرداندن خود شئ یک کپی از آن را ایجاد کرده و برگرداند. البته برای کلاس‌هایی مثل String که تغییر ناپذیر هستند، نیاز به این کار نیست (با این مورد در آینده بیشتر آشنا خواهید شد).
۶. از متدهای ستر[3] استفاده نمی‌کنیم.

به مثال صفحه بعد توجه کنید.

[1] Getter
[2] Primitive Type
[3] Setter

```java
class Student {

    private final String name;
    private final int regNo;
    private final String[] courses;
    public Student(String name, int regNo,String[] courses)
    {
        this.name = name;
        this.regNo = regNo;
        this.courses = courses;
    }

    public String getName() { return name; }

    public int getRegNo() { return regNo; }

    public String[] getCourses(){
        String[] tempCourses = new String[this.courses.length];
        for (int i = 0 ; i < tempCourses.length ; ++i){
            tempCourses[i] = this.courses[i];
        }
        return tempCourses;
    }
    public void printCourses(){
        for (String course : this.courses){
            System.out.println(course);
        }
    }
}
```

(پیاده‌سازی کلاس Student)

```java
public class Main {
    public static void main(String[] args) {

        String[] courses = {
                "AP","General Math II","Discrete Mathematics"
        };
        Student student1 = new Student("Alireza",101,courses);
        String[] studentCourses = student1.getCourses();
        studentCourses[1] = "General Physics II";
        student1.printCourses();
    }
}
```

(پیاده‌سازی کلاس Main)

خروجی مثال بالا:

```
AP
General Math II
Discrete Mathematics
```

(خروجی مثال قبل)

همان‌طور که مشاهده کردید، وقتی یک شئ از کلاس Student ساختیم، دیگر توانایی تغییر محتوای آن را نخواهیم داشت.

کلاس‌های تغییرناپذیر مزایای زیادی دارند. برای مثال، یک کلاس تغییرناپذیر برای برنامه‌نویسی موازی[1] که در آینده با آن آشنا خواهید شد، باعث آسان‌تر شدن طراحی برنامه می‌شود.

[1] Multithreaded Programming

کالکشن‌ها در جاوا

در این بخش با کالکشن‌ها در جاوا آشنا می‌شویم. کالکشن‌ها برای ذخیره و اداره کردن اشیاء استفاده می‌شوند. کالکشن‌های جاوا عملیات زیر را در اختیار ما قرار می‌دهند:

- Searching
- Sorting
- Insert
- Delete

در ادامه با برخی از کالکشن‌های جاوا آشنا می‌شویم:

آشنایی با ArrayList

پیشتر با مفهوم آرایه آشنا شدیم. اما آرایه نقاط ضعفی دارد، برای مثال اندازه‌ی آن هنگام تعریف باید مشخص باشد. یعنی نمی‌توانیم ۱۱ شئ را در یک آرایه به اندازه‌ی ۱۰ قرار دهیم. یکی از نقاط قوت arraylist که این مشکل را حل می‌کند، داشتن اندازه‌ی پویا[1] است.

برای استفاده از arraylist مانند تصویر زیر باید از کتابخانه‌ی java.util آن را ایمپورت[2] کنیم:

```
import java.util.ArrayList;
```

فرض کنید یک arraylist برای نگهداری نام چند ماشین نیاز داریم، مانند تصویر زیر یک شئ از کلاس ArrayList ایجاد می‌کنیم:

```
ArrayList<String> cars = new ArrayList<String>();
```

چند نکته در مورد کلاس داخل<>:

- در حالت کلی، می‌توان از کلاس‌هایی که خودمان نوشته‌ایم نیز استفاده کنیم. برای مثال اگر کلاسی به اسم Car داشتیم و در آن اطلاعات ماشین را ذخیره می‌کردیم، می‌توانستیم به جای <String>، <Car> قرار دهیم.
- در دستورکارهای گذشته با نوع‌های ابتدایی[3] آشنا شدیم. توجه داشته باشید که در arraylist نمی‌توان از آن‌ها استفاده کرد. راهکار استفاده از آن‌ها را در آینده و در مبحث wrapper class خواهید آموخت.

[1] Dynamic Size
[2] Import
[3] Primitive Types

در حال حاضر arraylist ما خالی است و اندازه‌ی آن صفر است. برای قرار دادن یک شئ در آن، از متد add استفاده می‌کنیم:

```
cars.add("BMW");
cars.add("Ford");
```

حال تصور کنید نیاز داریم شئ‌ای که در خانه‌ی اول arraylist قرار دارد را دریافت کنیم. برای این کار باید خانه‌ی با اندیس صفر را دریافت کنیم:

```
cars.get(0);
```

در مثال بالا، این دستور BMW را برمی‌گرداند.

در بعضی مواقع می‌خواهیم شئ‌ای که اضافه می‌کنیم، با شئ‌ای در درون arraylist جایگزین شود. برای این‌کار باید از از متد set استفاده نمود:

```
cars.set(0, "Opel");
```

دقت کنید که این دستور در مثال بالا، BMW را با Opel جایگزین می‌کند.

برای حذف کردن یک شئ از متد remove استفاده می‌کنیم:

```
cars.remove(0);
```

توجه داشته باشید که پس از حذف اندیس صفر که Opel بود، تمام اشیاء دیگر یک واحد به سمت چپ حرکت می‌کنند و در این مثال، Ford در اندیس صفر قرار می‌گیرد.

دو متد clear و size هم به ترتیب وظیفه‌ی پاک کردن تمام اشیاء و بدست آوردن اندازه‌ی arraylist را دارند:

```
cars.clear();
```

```
cars.size();
```

آشنایی با لینکدلیست[1]

قبل‌تر با لینکدلیست آشنا شدیم. جاوا یک کلاس آماده برای لینکدلیست دارد که متدهای آن شباهت زیادی به arraylist دارد.

توجه داشته باشید که با اینکه متدهای لینکدلیست و arraylist شباهت زیادی به یکدیگر دارند اما از لحاظ ساختاری با یکدیگر تفاوت‌هایی دارند و بهتر است با توجه به شرایط و کاری که قصد انجام آن را داریم، از یکی از آن‌ها استفاده کنیم. برای مطالعه‌ی بیشتر درباره‌ی ویژگی‌های لینکدلیست و arraylist و مزیت‌های آن‌ها نسبت به هم، می‌توانید به این لینک مراجعه کنید.

برای استفاده از لینکدلیست مانند تصویر زیر باید کلاس آن را ایمپورت کرد:

```
import java.util.LinkedList;
```

نوع تعریف کردن لینکدلیست مانند arraylist است و تمام متدهای arraylist که در قسمت بالا معرفی شد نیز در لینکدلیست وجود دارند:

```
LinkedList<String> cars = new LinkedList<String>();
cars.add("Volvo");
cars.add("BMW");
cars.add("Ford");
cars.add("Mazda");
cars.set(0, "Opel");
cars.get(0);
cars.remove( index: 0);
cars.size();
cars.clear();
```

آشنایی با هش‌ست[2]

از لحاظ متدها و طرز استفاده هش‌ست شباهت زیادی به arraylist و لینکدلیست دارد. اما با هم تفاوت‌هایی در استفاده و ساختار دارند:

- در هش‌ست تمام اشیاء متمایز و نمی‌توان یک شئ را چند بار اضافه نمود (مانند مجموعه‌ها در ریاضی).
- در هش‌ست، اشیاء ترتیب ندارند و در نتیجه نمی‌توان از اندیس استفاده کرد.

برای استفاده از هش‌ست مانند تصویر زیر باید کلاس آن را ایمپورت کنیم:

```
import java.util.HashSet;
```

[1] Linked List
[2] Hash Set

سپس هش‌ست را مانند تصویر زیر تعریف می‌کنیم:

```
HashSet<String> cars = new HashSet<String>();
```

در ادامه با چند متد از کلاس هش‌ست بیشتر آشنا می‌شویم. متد add برای اضافه کردن شئ به هش‌ست:

```
cars.add("Volvo");
cars.add("BMW");
cars.add("Ford");
cars.add("BMW");
cars.add("Mazda");
```

توجه کنید با وجود اینکه شیء BMW دو بار اضافه شده، اما مقادیر داخل هش‌ست به دلیل خاصیت متمایز بودن اشیاء به صورت زیر خواهد بود:

```
[Volvo, Mazda, Ford, BMW]
```

برای حذف یک شئ از متد remove استفاده می‌کنیم:

```
cars.remove("Volvo");
```

متد contains در تمام مقادیر موجود در هش‌ست جستجو می‌کند و اگر آن شئ در هش‌ست وجود داشت مقدار trueو در غیر این صورت مقدار false را برمی‌گرداند:

```
cars.contains("Mazda");
```

دو متد clear و size که در arraylist با آنها آشنا شدید در کلاس هش‌ست نیز وجود دارند.

توجه داشته باشید که برای پیمایش روی هش‌ست با توجه به نبود اندیس برای اشیاء، می‌توان از for-each استفاده کرد:

```
for (String i : cars) {
  System.out.println(i);
}
```

همچنین قاعده‌ی <> که در arraylist توضیح داده شد، در هش‌ست نیز برقرار است. با این تفاوت که برای استفاده از کلاس‌هایی که توسط خودمان ایجاد شده است، باید متدهای hashcode و equals را در آن‌ها ایجاد کنیم (برای انجام این کار در اینتلیجی[1] می‌توان در کلاس مورد نظر از میانبر alt + insert استفاده کرد).

آشنایی با هش‌مپ[2]

هش‌مپ کمی با سه کالکشن قبلی متفاوت است. در هش‌مپ تعدادی از جفت‌های کلید[3] و مقدار[4] داریم که کلید به مقدار تناظر داده شده است و با داشتن کلید، می‌توان مقدار را دریافت کرد. همچنین در هش‌مپ همانند یک تابع، کلیدها باید منحصر به فرد باشند اما دو کلید متفاوت می‌توانند به مقدار یکسانی تناظر داده شوند. علاوه بر آن، در هش‌مپ مانند هش‌ست، ترتیب وجود ندارد و در نتیجه کلیدها و مقدارها اندیس ندارند.

برای استفاده از هش‌مپ مانند تصویر زیر باید کلاس آن را از کتابخانه‌ی مربوطه ایمپورت کرد:

```
import java.util.HashMap;
```

سپس هش‌مپ را مانند تصویر زیر تعریف می‌کنیم:

```
HashMap<String, String> capitalCities = new HashMap<String, String>();
```

توجه کنید که نوع کلید سمت چپ و نوع مقدار سمت راست قرار می‌گیرد.

برای اضافه کردن یک جفت (کلید، مقدار) در هش‌مپ از متد put استفاده می‌کنیم که دو ورودی دارد:

```
capitalCities.put("England", "London");
capitalCities.put("Germany", "Berlin");
```

برای دریافت مقدار مربوط به یک کلید از متد get استفاده می‌کنیم:

```
capitalCities.get("England");
```

در این مثال، London بازگردانده می‌شود.

برای پاک کردن یک جفت (کلید و مقدار)، کلید مربوط به آن را به متد remove ورودی می‌دهیم:

```
capitalCities.remove("England");
```

---

[1] IntelliJ
[2] HashMap
[3] Key
[4] Value

دو متد clear و size که در arraylist با آن‌ها آشنا شدیم، در کلاس هش‌مپ نیز وجود دارند.

در هش‌مپ نیز مانند هش‌ست به دلیل نداشتن ترتیب، برای پیمایش روی مقادیر کلیدها و مقدارها می‌توان از for-each استفاده کرد.

پیمایش روی کلیدها:

```
for (String i : capitalCities.keySet()) {
  System.out.println(i);
}
```

پیمایش روی مقدارها:

```
for (String i : capitalCities.values()) {
  System.out.println(i);
}
```

همچنین قاعده‌ی <> که در arraylist توضیح داده شد، در هش‌مپ نیز برقرار است. با این تفاوت که برای استفاده از کلاس‌هایی که توسط خودمان ایجاد شده است، باید متدهای hashcode و equals را در آن‌ها ایجاد کنیم (برای انجام این کار در اینتلیجی می‌توان در کلاس مورد نظر از میانبر alt + insert استفاده کرد).

توجه داشته باشید که در این قسمت، هدف تنها آشنا شدن با داده‌ساختارهای معروف و پیاده‌سازی آنها در جاوا بود. در ترم‌های آینده و در درس ساختمان‌داده‌ها و الگوریتم‌ها به صورت دقیق‌تر با آن‌ها آشنا خواهید شد.

آشنایی با Iteratorها

معرفی Iterator

فرض کنید یک arrayList از String داریم که می‌خواهیم اعضای آن که شامل کلمه Java هستند را حذف کنیم.
در این مواقع به Iterator نیاز داریم.
به مثال زیر توجه کنید:

```
import java.util.ArrayList;
import java.util.Iterator;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> strings = new ArrayList<>();

        strings.add("Java SE");
        strings.add("Java ME");
        strings.add("C Programing");
        strings.add("Python");

        for (String s : strings) System.out.println(s);


System.out.println("*********************************************");
        Iterator<String> it = strings.iterator();

        while (it.hasNext()){
            String str = it.next();
            if (str.contains("Java")){
                it.remove();
            }
        }

        for (String s : strings) System.out.println(s);
    }
}
```

(نمونه‌ی استفاده از Iterator)

خروجی:

```
Java SE
Java ME
C Programing
Python
*********************************************
C Programing
Python
```

(خروجی مثال بالا)

Iterator برای هش‌ست، هش‌مپ، لینکدلیست و غیره قابل استفاده است. با موارد دیگر استفاده از Iterator در ادامه بیشتر آشنا خواهید شد.

انجام دهید: سیستم رأی‌گیری

از شما خواسته شده است تا یک نرم‌افزار رأی‌گیری با استفاده از زبان جاوا بنویسید. در این نرم‌افزار، کاربر می‌تواند یک رأی‌گیری ایجاد کرده و پس از ساختن آن، سایر افراد می‌توانند آرای خود را ثبت نمایند. رأی‌گیری می‌تواند دارای دو مدل باشد:

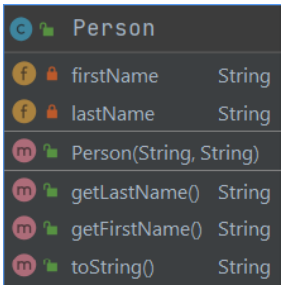
۱. هر فرد تنها بتواند یک رأی بدهد.
۲. هر فرد بتواند چندین رأی بدهد.

همچنین رأی‌گیری می‌تواند به صورت ناشناس باشد، یعنی در هنگام اعلام نتایج، رأی هر فرد برای بقیه نشان داده نشود. برای پیاده‌سازی این بخش، از طراحی‌ای که در ادامه در اختیار شما قرار می‌گیرد استفاده نمایید.
فراموش نکنید که کدهای خود را بر روی یک ریپازیتوری با نام «AP-Workshop4» قرار دهید.

کلاس Person

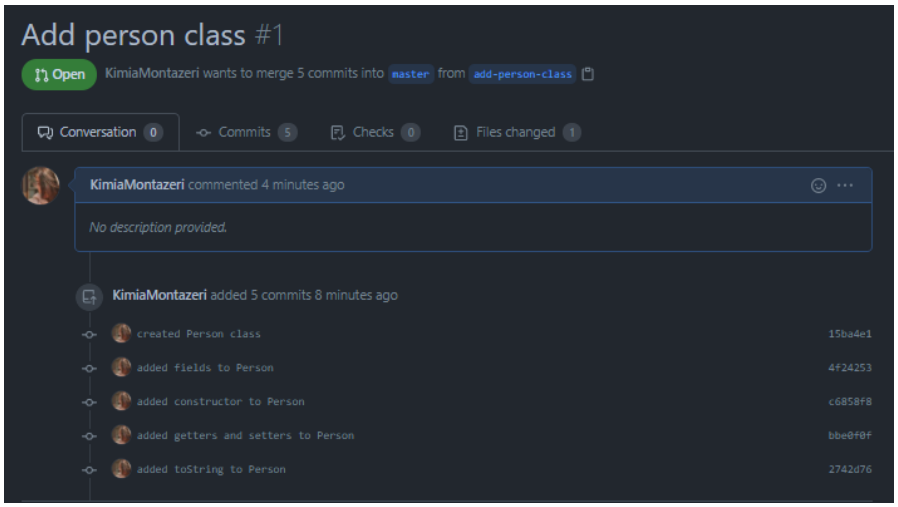
این کلاس شامل فیلدهای firstname و lastname می‌باشد:

- `private String firstName;`
- `private String lastName;`



(ساختار کلاس Person)

تحویل این بخش: این کلاس را در برنچ جدا ساخته؛ سپس فیلدها، گتر، ستر و کانستراکتورها و متدهای دیگر را به آن اضافه کنید و برای هر کدام، کامیت مناسب بزنید. در آخر pull request داده و این برنچ را با master مرج کنید:



(تحویل بخش اول)

کلاس Vote

هر رأی شامل یک رأی‌دهنده و تاریخ رأی‌دهی می‌باشد. پس این موارد را به صورت فیلد در این کلاس قرار می‌دهیم (بهتر است آن‌ها را final تعریف کنیم، چون نیازی به تغییر دادن آنها نداریم):

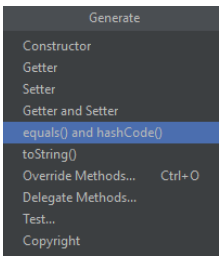
- `private final Person voter`
- `private final String date`

حال constructor و getterهای را اضافه می‌کنیم:

- `public Vote(Person voter, String date)`
- `public Person getVoter()`
- `public String getDate()`

متدهای equals و hashCode را تشکیل دهید (در ادامه به دلیل وجود این متدها پی خواهید برد).

برای انجام این کار در اینتلیجی، با استفاده از میانبر fn + alt + delete، منوی generate نمایش داده خواهد شد:



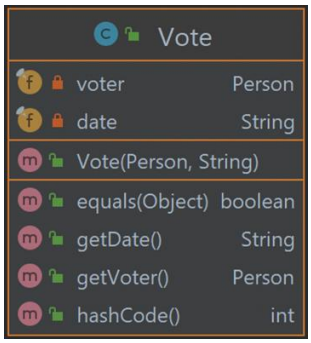
(منوی generate)

در هنگام ساخت این متدها، تمام فیلدهای کلاس را انتخاب کنید. در نهایت، پیاده سازی متدها به صورت زیر خواهد بود:

```java
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof VotingSystem)) return false;
    VotingSystem that = (VotingSystem) o;
    return Objects.equals(getVotingList(), that.getVotingList());
}

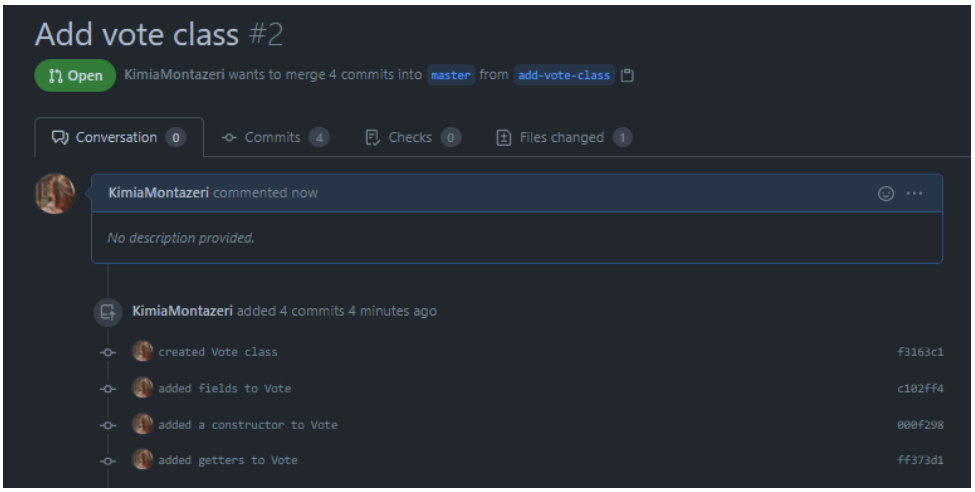
@Override
public int hashCode() {
    return Objects.hash(getVotingList());
}
```

(پیاده‌سازی متدهای equals و hashCode)



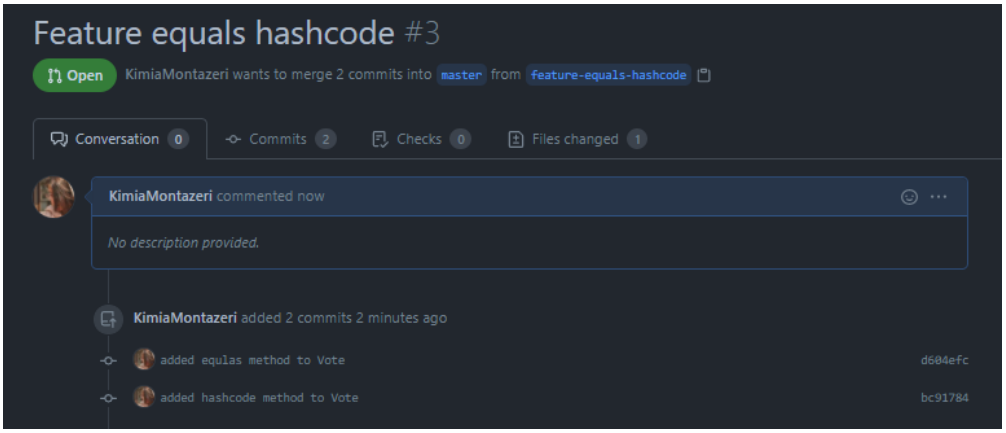
(ساختار کلاس Vote)

تحویل این بخش: برای تشکیل کلاس و اضافه کردن فیلدها، گتر، ستر و کانستراکتورها، برنچ جدیدی بسازید و پس از کامل شدن، آن را با برنچ master مرج کنید:



(ساخت برنچ برای تشکیل کلاس‌ها)

در برنچ دیگر، متدهای equals و hashCode را تشکیل دهید و پس از کامل شدن، آن را با برنچ master مرج کنید:



(ساخت برنچ برای پیاده‌سازی متدهای equals و hashCode)

کلاس Voting

حالت رأی‌گیری (فیلد type)، تک رأی و چند رأی (۰ به معنای تک‌رأی و ۱ به معنای چندرأی) بودن را مشخص می‌کنند:

- `private int type;`

پرسش رأی‌گیری (فیلد question):

- `private String question;`

گزینه‌های رأی‌گیری (در هش‌مپ فیلد choices هر String نشان دهنده‌ی یک گزینه رأی‌گیری می‌باشد که به یک هش‌ست از رأی‌های داده شده به آن تناظر داده شده است):

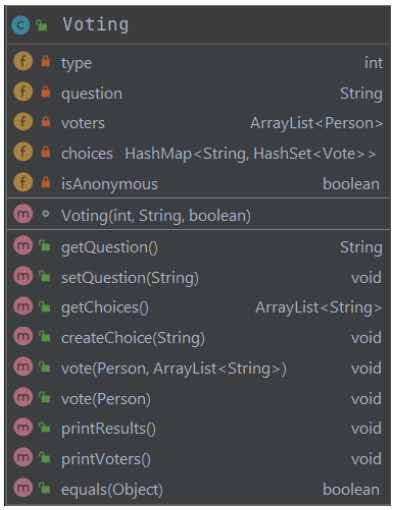
- `private HashMap<String, HashSet<Vote>> choices;`

ناشناس بودن رأی‌گیری (فیلد isAnonymous):

- `private boolean isAnonymous;`

مجموعه‌ی رأی‌دهنده‌ها (فیلد voters):

- `private ArrayList<Person> voters;`



(ساختار کلاس Voting)

توضیحات متدها

متد getChoices: گزینه‌های رأی‌گیری را در یک ArrayList برمی‌گرداند:

- `public ArrayList<String> getChoices()`

متد createChoices: یک رشته را در پارامتر ورودی خود دریافت می‌کند و آن را به گزینه‌های رای‌گیری اضافه می‌کند:

- `public void createChoice(String choice)`

متد vote: متد vote اول برای حالتی است که رأی‌گیری به صورت ناشناس نیست و پارامتر ورودی این متد شامل رأی‌دهنده و گزینه‌های انتخاب شده‌ی وی است. متد vote دوم (اورلود[1] شده) برای حالت رأی‌گیری ناشناس می‌باشد و باید یک گزینه تصادفی تولید و انتخاب شود:

- `public void vote(Person voter, ArrayList<String> voter_choices)`
- `public void vote(Person person)`

متد printResults: نتیجه‌ی رأی‌گیری (شامل گزینه‌های رأی‌گیری و تعداد رأی هر گزینه می‌باشد) چاپ شود:

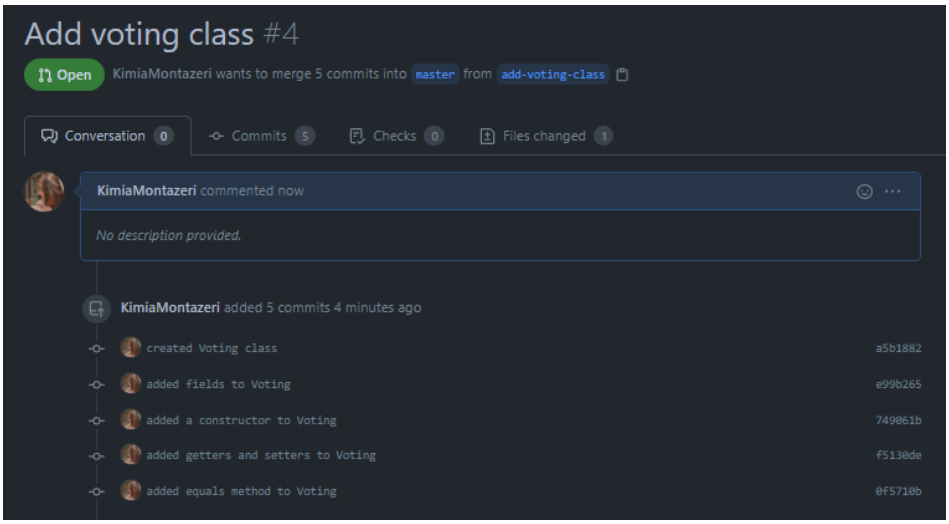
- `public void printResults()`

متد printVoters: در صورتی که رأی‌گیری ناشناس نبود، گزینه‌های رأی‌گیری و تمامی افرادی که به آن گزینه رأی داده‌اند، چاپ می‌شود:

- `public void printVoters()`

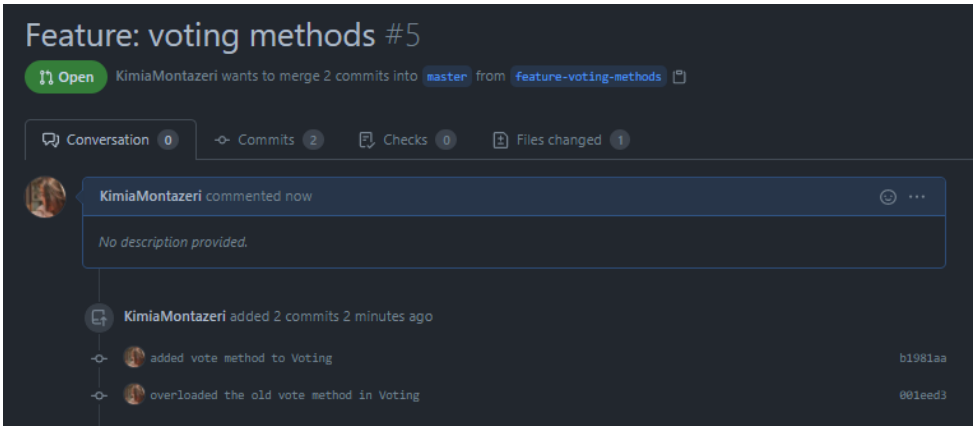
[1] Overload

تحویل این بخش: به علت بزرگ بودن این کلاس، می‌توان روند ساخت آن را به مراحل جزئی‌تری تقسیم کرد. ابتدا برای تشکیل کلاس و اضافه کردن فیلدها، گتر، ستر و کانستراکتورها، برنچ جدیدی بسازید و پس از کامل کردن، آن را با برنچ master مرج کنید:



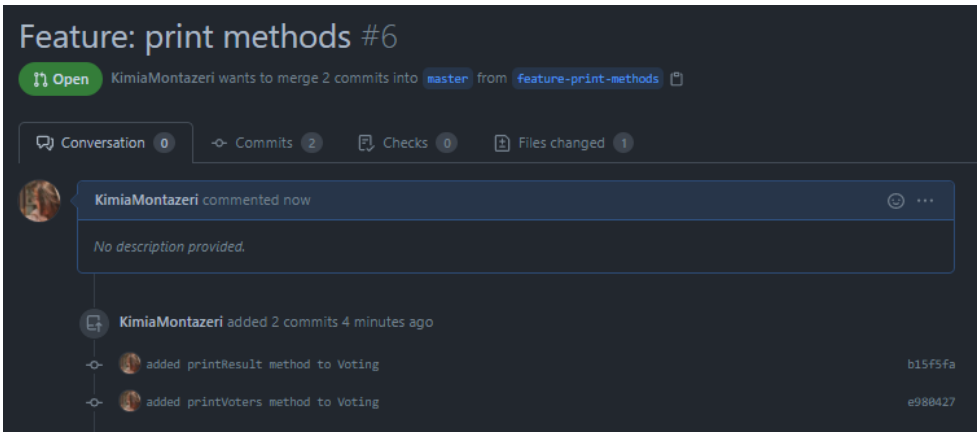
(ساخت برنچ برای پیاده‌سازی گتر و سترها)

سپس متدهای vote را در برنچ دیگر تشکیل دهید:



(ساخت برنچ برای پیاده‌سازی متد vote)

پس از آن، در یک برنچ جدید، متدهای printVoters و printResults را اضافه کنید:

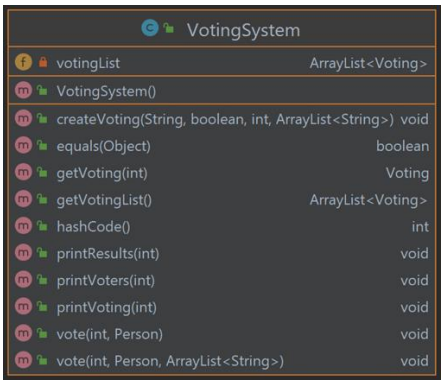


(ساخت برنچ برای پیاده‌سازی متدهای printVoters و printResults)

کلاس VotingSystem

این کلاس وظیفه‌ی مدیریت تمام رأی‌گیری‌های ساخته شده را دارد. پس باید لیست تمام رأی‌گیری‌ها در آن قرار گیرد:

- `private ArrayList<Voting> votingList`



(ساختار کلاس VotingSystem)

توضیحات متدها

متد createVoting: با توجه به آرگومان‌های داده شده، یک رأی‌گیری جدید می‌سازد:

- ```
  public void createVoting(String question,
                           boolean isAnonymous,
                           int type,
                           ArrayList<String> choices)
  ```

متد getVoting: با توجه به آرگومان index، رأی‌گیری موردنظر را در votingList پیدا می‌کند:

- `public Voting getVoting(int index)`
  ```

متد getVotingList: متد گتر برای فیلد votingList است:

- `public ArrayList<Voting> getVotingList()`

متد printResults: نتیجه‌ی نهایی رأی‌گیری با شماره‌ی index را چاپ می‌کند:

- `public void printResults(int index)`

متد printVoters: لیست رأی‌دهندگان را برای رأی‌گیری با شماره‌ی index را چاپ می‌کند:

- `public void printVoters(int index)`

متد printVoting: سؤال‌ها و گزینه‌های موجود در رأی‌گیری با شماره‌ی index را چاپ می‌کند:

- `public void printVoting(int index)`

رأی‌دهی تصادفی

یکی از راه‌های پیاده‌سازی این قسمت، اورلود کردن متد vote (که در بالا توضیح داده شد) می‌باشد:

- `public void vote(int index, Person voter)`

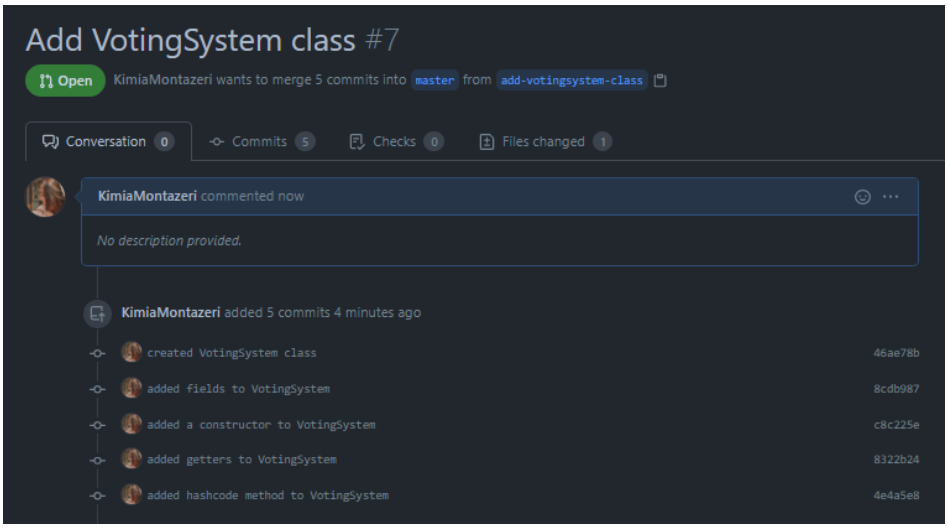
برای تولید مقادیر تصادفی، از کلاس Random در کتابخانه‌ی java.util استفاده می‌کنیم:

```
Random r = new Random();
r.nextInt(); // generates a random number
r.nextInt( bound: 10); // generates a random number in [0,10)
```

(نحوه‌ی تولید مقادیر تصادفی)

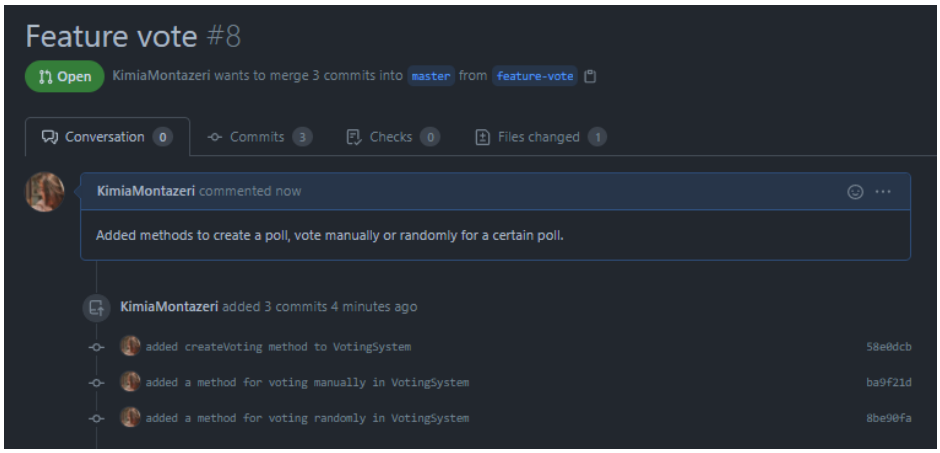
برای آشنایی بیشتر با این کلاس، می‌تون به این لینک مراجعه کرد.

تحویل این بخش: برای تشکیل کلاس و اضافه کردن فیلدها، گتر، ستر و کانستراکتورها، برنچ جدیدی بسازید و پس از کامل شدن، آن را با برنچ master مرج کنید:



(ساخت برنچ برای پیاده‌سازی گتر و سترها)

برای اضافه کردن فیچر رأی‌دهی، برنچ جدیدی ساخته و در آن متدهای لازم را تشکیل دهید:



(ساخت برنچ برای پیاده‌سازی فیچر رأی‌دهی)

در آخر، سه متد مربوط به print را در یک برنچ دیگر اضافه کنید.

کلاس Main

در برنچ جدیدی این کلاس را کامل کنید و در آخر pull request بدهید.

تست برنامه

۱. در کلاس Main یک شیء از کلاس Voting System ساخته و چند شئ Voting با نوع‌های مختلف به آن اضافه کنید.
۲. سپس چند Person ساخته و از طریق آن‌ها، به رأی‌گیری‌های مختلف رأی دهید.
۳. نتایج رأی‌گیری‌ها را در کنسول نمایش دهید.