به نام خدا

دستور کار کارگاه برنامه‌نویسی پیشرفته

جلسه دوم

آشنایی با مفاهیم برنامه‌نویسی

مقدمه

در این جلسه قصد داریم تا با استفاده از مفاهیم کلاس و شی در زبان جاوا یک کلاس را به همراه دانشجوهای آن مدل کنیم. در کنار کار با جاوا می‌خوایم با ابزار git نیز مقداری آشنا شویم که ابزار کارآمدی برای توسعه‌ی بسیاری از نرم‌افزارها است. علاوه بر ابزار git و برنامه‌نویسی شی‌گرا، قصد داریم تا با مفهوم توسعه آزمون محور[1] نیز آشنا شویم که باعث افزایش میزان کیفیت محصول نرم‌افزاری می‌شود.

ابزار کنترل نسخه Git

توسعه و نگهداری نرم‌افزار کاری گروهی و مستمر در طول زمان است. از آنجایی که محصولات نرم‌افزاری در معرض تغییر هستند و نسخه‌های جدید محصول باید تولید شود، کدهای نوشته شده نیز همواره در معرض توسعه، تغییر و حذف توسط این تیم نرم‌افزاری هستند. از این رو وجود یک ابزار مدیریت تغییرات کد برای تیم‌های توسعه نرم‌افزار بسیار لازم است.

یکی از این ابزارهای کنترل نسخه که مدیریت تغییرات را بر عهده می‌گیرد، ابزار git است. این ابزار اولین بار توسط Linus Torvalds برای توسعه هسته سیستم‌عامل لینوکس بکار رفت[2]. هم اکنون git به یکی از موفق‌ترین و مهمترین ابزارهای مدیریت نسخه تبدیل شده و یادگیری آن برای هر برنامه‌نویسی لازم است. در ادامه دستورکار قصد داریم تا با استفاده از این ابزار، تغییرات رخ داده در محصول خود را مدیریت کنیم.

نصب و راه‌اندازی Git

در ابتدا نرم‌افزار git را از لینک زیر دانلود کنید و نرم‌افزار را نصب کنید.

[1] Test Driven Development

[2] https://en.wikipedia.org/wiki/Git

https://git-scm.com/download/win

نرم‌افزار IntelliJ را باز کنید و یک پروژه جاوا بسازید. دقت کنید که تیک create project from template را نزنید. سپس نرم‌افزار git bash را اجرا کنیدو به پوشه‌ای که پروژه را ساخته اید بروید.

```
Alireza@DESKTOP-VURISBU MINGW64 ~
$ cd Desktop/testGit/
```

شکل 1 دستور ورود به پوشه پروژه

سپس دستور git init را وارد کنید. با وارد کردن این دستور یک پوشه به نام git. به پوشه پروژه شما اضافه می‌شود. در این پوشه اطلاعات مربوط به تغییرات پروژه ذخیره می‌شود.

```
Alireza@DESKTOP-VURISBU MINGW64 ~/Desktop/testGit
$ git init
Initialized empty Git repository in C:/Users/Alireza/Desktop/testGit/.git/
```

شکل 2 دستور git init

با دستور git status می‌توانید وضعیت تغییرات خود را از زمان آخرین commit (در ادامه با این واژه آشنا خواهید شد) تا کنون مشاهده کنید. خروجی این دستور در شکل 3 نمایان است. همانطور که مشاهده می‌کنید بعد از وارد کردن دستور عبارت‌های زیر نوشته شده:

on branch master:

با معنای این عبارت در جلسات بعدی آشنا خواهید شد.

untracked files:

در اینجا لیست فایل‌ها و پوشه‌هایی را نشان می‌دهد که بوسیله‌ی گیت کنترل نمی‌شوند و تغییرات آن‌ها در پوشه git. ثبت نمی‌شود. بعضی از فایل‌ها و پوشه‌ها مانند فایل تنظیمات نباید در git ثبت شود چرا که ممکن است این فایل‌ها مختص محیط خاصی باشند و توسعه دهندگان دیگر تنظیمات خود را داشته باشند. از این رو ابزار گیت این قابلیت را به کاربر می‌دهد تا خود بصورت دستی فایل‌ها و پوشه‌ها را به گیت اضافه کند.

```
Alireza@DESKTOP-VURISBU MINGW64 ~/Desktop/testGit (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .idea/
        testGit.iml

nothing added to commit but untracked files present (use "git add" to track)
```

شکل 3 خروجی دستور git status

از آنجا که پوشه‌ی /idea. و فایل testGit.iml برای تنظیمات پروژه است نمیخواهیم این فایل‌ها را به git اضافه کنیم. در ادامه به نوشتن یک برنامه ساده در جاوا می‌پردازیم.

مدل‌سازی آزمایشگاه درس

در این قسمت می‌خواهیم با استفاده از مفاهیم کلاس و شئ که در درس با آنها آشنا شدید یک دانشجو را مدل کنیم. برای اینکه کدهایمان خوانا باشند، به نحوه تعریف کلاس‌ها، شئ‌ها، متدها، متغیرها، چینش متدها و متغیرها در یک کلاس دقت کنید. تمام این موارد را با مثال خواهیم دید.

مراحل انجام کار

در ابتدا یک کلاس Student در فایل Student.java ایجاد کنید. از آنجایی که در پروژه تغییری داده‌ایم میخواهیم آنرا در git ثبت کنیم. برای این کار دوباره دستور git status را وارد کنید. شکل 4 خروجی دستور را نمایش می‌دهد. همانگونه که پیدا است این دفعه پوشه src به لیست پوشه‌ها و فایل های دنبال نشده (untracked) اضافه شده. از آنجایی که کدهای ما داخل این پوشه است پس باید این پوشه را به لیست پوشه‌هایی که git باید نسخه های آنرا مدیریت کند اضافه کنیم. دستور زیر این کار را انجام می‌دهد.

git add <file-name>

که <file-name> نام فایل یا پوشه‌ای است که قصد track کردن آنرا دارید.

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .idea/
        src/
        testGit.iml

nothing added to commit but untracked files present (use "git add" to track)
```

شکل 4خروجی دستور git status

در اینجا دستور زیر را وارد کنید.

git add src

سپس دستور git status را دوباره وارد کنید. در خروجی دستور این بار یک عبارت جدید مشاهده می‌شود.

changes to be committed:

در اینجا لیست فایل‌هایی نشان داده می‌شود (با رنگ سبز) که از زمان آخرین commit تا کنون تغییر کرده اند و تغییرات آنها باید ثبت یا commit شود.

```
Alireza@DESKTOP-VURISBU MINGW64 ~/Desktop/testGit (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   src/Student.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .idea/
        testGit.iml
```

شکل 5 خروجی دستور git status

حال با دستور git commit می‌خواهیم تغییرات خود را ذخیره کنیم. فرمت این دستور بصورت زیر است:

git commit –m <commit-message>

که در آن commit-message پیامی است که بیانگر تغییرات کد است. commit کردن همانند این است که به git بگویید این تغییرات انجام شده در کد را با یک سری مشخصات ثبت کن. این مشخصات شامل تاریخ commit، پیامی که نویسنده کدها برای ثبت آن انتخاب کرده. نام کسی که تغییرات را اعمال کرده (در اینجا چون فقط خود شما کدها را تغییر می‌دهید فقط نام خودتان را مشاهده می‌کنید) و یک عبارت hash شده است.

در این حالت باید عبارت زیر را وارد کنیم که به این معنا است که ما کلاس student را به پروژه اضافه کردیم:

Student class created.

پس دستور زیر را وارد کنید

git commit –m “Student class created.”

در صورت مشاهده خطا شکل 6 دستورات زیر را وارد کنید:

git config user.email “your-email”

git config user.name “your-name”

سپس دوباره دستور git commit را وارد کنید.

```
Alireza@DESKTOP-VURISBU MINGW64 ~/Desktop/testGit (master)
$git commit -m "Student class created"

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'Alireza@DESKTOP-VURISBU.(none)')
```

شکل 6 خطای دستور commit

```
Alireza@DESKTOP-VURISBU MINGW64 ~/Desktop/testGit (master)
$git conmmit -m "Student class created"
[master (root-commit) 97773cc] class student created
 1 file changed, 2 insertions(+)
 create mode 100644 src/Student.java
```

شکل 7 خروجی دستور git commit

حال اگر دوباره دستور git status بزنیم خروجی زیر را مشاهده خواهیم کرد.

```
Alireza@DESKTOP-VURISBU MINGW64 ~/Desktop/testGit (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .idea/
        testGit.iml

nothing added to commit but untracked files present (use "git add" to track)
```

شکل 8 خروجی دستور git status بعد از git commit

حال می‌خواهیم توابع و متغیرهای کلاس Student را بنویسیم. در ابتدا پارامترهای کلاس را مطابق قطعه کد زیر وارد کنید.

```
public class Student {
        private String firstName;
        private String lastName;
        private String studnetID;
        private float grade;
}
```

سپس متدهای آنرا اضافه می‌کنیم. برای هر کلاس، در ابتدا فقط نام متدها و ورودی آنها را می‌نویسیم. برای آنکه کدها دچار خطای کامپایل نشوند، خروجی هر تابع را باید return کرد. پس در ابتدا از مقادیر پیش فرض استفاده می‌کنیم.

```
public void setFirstName(String firstName){}
public void setLastName(String lastName){}
public void setStudentID(String ID){}
public Student(String firstName, String lastName, String studnetID){}
public void setGrade(float grade){}
public float getGrade(){
```

```
        return 0;
}
public String getFirstName(){
        return "";
}
public String getLastName(){
        return "";
}
public String getStudentID(){
        return "";
}
public String getInfo(){
        return "";
}
```

حال تغییرات خود را دوباره commit کنید. برای این کار ابتدا خروجی دستور git status را مشاهده کنید.

```
Alireza@DESKTOP-VURISBU MINGW64 ~/Desktop/testGit (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/Student.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .idea/
        testGit.iml

no changes added to commit (use "git add" and/or "git commit -a")
```

شکل 9 خروجی دستور git status بعد از پیاده سازی توابع کلاس Student

همانطور که مشاهده می‌کنید تغییراتی نیاز است که ثبت شود و لیست فایل‌هایی که باید تغییر کند آورده شده.

سپس با دستور git commit –m “Methods of Student class created.” تغییرات را ثبت کنید.

توسعه آزمون محور

یکی از روش‌های توسعه نرم‌افزار، توسعه‌ی آزمون محور است. در این روش ابتدا تیم توسعه‌گر یک سری تست آماده می‌کنند و سپس به توسعه‌ی محصول خود می‌پردازند. هدف از نوشتن تست‌ها این است

برنامه نویس‌ها بتوانند کلاس‌ها، متغیرها و متدهای خود را مطابق با خواسته‌های پروژه و بطور درست پیاده‌سازی و ارزیابی کنند. از این رو نوشتن تست در پروژه‌ها بسیار اهمیت دارد.

در این قسمت یک فایل تست به نام StudentTest.java قرار داده شده است. آن‌را دانلود و به پروژه اضافه کنید. سپس این فایل را اجرا کنید. مشاهده خواهید کرد که تمامی تست‌ها جواب false به شما می‌دهند. در قسمت بعدی باید متدهای کلاس Student را پیاده کنید تا تمامی جواب‌ها true شود. در این حالت کد شما پذیرفته خواهد شد.

انجام دهید:

در این قسمت وظیفه‌ی هر تابع را می‌نویسیم.

توابع setFirstName و getFirstName برای مقداردهی و خواندن متغیر firstName هستند.

توابع setLastName و getLastName برای مقداردهی و خواندن متغیر lastName هستند.

توابع setGrade و getGrade برای مقداردهی و خواندن متغیر grade هستند.

تابع getInfo نیز مشخصات دانشجو را بصورت زیر در قالب string بازمی‌گرداند:

Name: <firstName> <lastName>

Student ID: <studentID>

Grade: <grade>

توجه کنید که فاصله گذاری ها (گذاشتن space ها) همانند فرمت بالا باشد تا خروجی درست شود.

تغییرات خود را با پیام "Student class completed." commit کنید.

تعامل کلاس‌ها

در این قسمت می‌خواهیم تعامل بین دو کلاس را مشاهده کنیم. یک کلاس به نام Lab ایجاد کنید شامل متدهای زیر:

```
public class Lab {
        private Student[] studnets;
        private int dayOfTheWeek;
        private int currentCapacity;
        public Lab (int dayOftheWeek, int capacity){}
        public void enrollStudent(Student std){}
        public Student[] getStudents(){}
        public float getAverage(){}
```

```
        public int getCurrentCapacity(){}
        public int getDayOfTheWeek(){}
}
```

انجام دهید:

متدهای داده شده کلاس Lab را پیاده‌سازی کنید. شرح هر متد بصورت زیر است

متدهای set و get برای نوشتن و خواندن پارامترهای کلاس هستند.

متد getAverage میانگین نمره‌های دانشجویان را حساب می‌کند و بازمی‌گرداند.

تابع enrollStudnet نیز اینگونه است که بررسی می‌کند در صورتی که currentCapacity از مقدار capacity کمتر باشد، دانشجو را به لیست دانشجویان کلاس اضافه کرده و currentCapacity را بروز می‌کند (یکی به آن اضافه می‌کند).

سپس فایل تست نوشته شده برای Student را نگاه کرده و سعی کنید یک فایل تست برای کلاس Lab بنویسید (حداقل 3 متد آنرا تست کنید).

در نهایت نیز تغییرات خود را commit کنید.

با دستور git log و git log –stat و git log –p روند commit های خود را مشاهده کنید.

تمرین برنامه‌نویسی اول

این قسمت برای تمرین در خانه است و کدهای آن باید در quera آپلود و داوری شود. تاریخ تحویل آن در quera مشخص شده است.

1-مین‌یاب:

در این سوال می‌خواهیم بازی مین‌یاب را تا حدی پیاده‌سازی کنیم. برنامه ای بنویسید که ابتدا به عنوان ورودی m و n را از کاربر بگیرد سپس اطلاعات یک ماتریس m*n از نوع کاراکتر را از کاربر بگیرد. خانه‌های مشخص شده با ’0’ خانه‌های خالی و خانه‌های مشخص شده با ’*’ نشان دهنده ی خانه ی دارای مین است.

به عنوان مثال به ورودی زیر توجه کنید:

```
10
```

```
5
*00000000*
000000*000
00000000**
000***00*0
000*000000
```

خروجی برنامه یک ماتریس m*n است که برای خانه‌های خالی متناظر ورودی تعداد بمب های اطراف و برای خانه‌های دارای مین شامل '*' است.

برای ورودی مثال بالا خروجی به صورت زیر میباشد:

```
*10001111 *
110001*233
00123323 **
002***12*3
002*421111
```

2-انتگرال گیری:

برنامه ای بنویسید که یک عبارت جبری را از ورودی دریافت کرده و انتگرال آن را به صورت یک رشته در خروجی چاپ کند.

فرمت کلی هر جمله از عبارت به صورت {power}x^{sign} میباشد که طبیعتاً ممکن است قسمت هایی از آن در جمله موجود نباشد.

به عنوان مثال ورودی زیر را در نظر بگیرید

```
4x^2-5x^4+x-2
```

خروجی

```
x^5+1.33x^3+0.5x^2-2x
```

نکات:

1) ضریب جملات رشته ورودی عدد صحیح و توان آنها اعداد صحیح نامنفی اند.

2) همانطور که می دانید جملات با ضریب صفر، توان 1، علامت مثبت جمله اول، ضریب 1، x با توان صفر و... نباید در رشته خروجی درج گردند. (یعنی رشته خروجی طبق استاندارد ریاضی باید ساده شده باشد)

3) ضریب جملات خروجی به صورت عدد اعشاری تا دو رقم اعشار (شبیه نمونه) در رشته قرار میگیرند. (صفرهای بی ارزش بعد از اعشار نباید چاپ گردند)

4) عبارت جبری حاصل را تا حد امکان ساده کنید (مثلا به جای x^5+x^5 باید 2x^5 نوشته شود)

5) عبارات نهایی باید به ترتیب درجه x در رشته خروجی مرتب گردند.

6) در پایان ضمن عذرخواهی از ریاضی دانان عزیز، نیازی به جمع کردن ثابت C با حاصل انتگرال نمیباشد.

3-بزرگترین زیررشته مشترک:

برنامه ای بنویسید که ابتدا عدد صحیح n و پس از آن n رشته را از ورودی بگیرد . خروجی برنامه بزرگترین رشته ای مانند S خواهد بود که هر کدام از رشته ها، S و یا وارون آن را به عنوان زیر رشته داشته باشند. اگر زیر رشته ی مشترکی وجود نداشت، چیزی چاپ نشود.

زیر رشته ای که در خروجی چاپ میشود ، باید به فرمی باشد که در رشته ی اول قرار دارد. مثلا در مثال زیر، باید FEDC چاپ شود ، و نه CDEF

نمونه ورودی

```
3
ABCDEF
FEDCAB
GHCDEFJK
```

خروجی آن

```
CDEF
```

4-پرانتزهای بالانس[3]:

در این سوال یک عبارت پرانتزی به عنوان ورودی داده می‌شود که شامل دو کاراکتر "(" و ")" است. هنگامی این عبارت بالانس است که پرانتز گذاری آن بصورت صحیح صورت گرفته باشد. پرانتزگذاری صحیح عبارت است از:

- هر پرانتزی که باز شده است بسته شده باشد.

[3] Balanced parentheses

- هنگامی که پرانتزی بسته می‌شود، پرانتز باز آن باید قبل از آن حضور داشته باشد.

ورودی مساله یک خط است که عبارت پرانتزها است. خروجی مساله باید بزرگترین زیر رشته بالانس پیوسته‌ای باشد که از ابتدای رشته شروع شده.

مثال 1:

ورودی

```
() () ()
```

خروجی

```
() () ()
```

مثال 2:

ورودی

```
( () () ) () ( ( () ) )
```

خروجی

```
( () () ) () ( ( () ) )
```

مثال 3:

ورودی

```
( () () ) ) (
```

خروجی

```
( () () )
```

مثال 4 :

ورودی

```
( () () ) ( ( ( ( ()
```

خروجی

```
( () () )
```

نکته: قبل از اینکه شروع کنید به کد زدن، پیشنهاد می‌کنیم روی راه حل فکر و تحقیق کنید.