

آموزش Git

مقدمه

اگر تا کنون تجربه‌ای هرچند کوتاه در برنامه‌نویسی داشته باشید احتمالاً تجربه‌ی از دست‌دادن بخشی از پروژه را به دلایل گوناگون مثل ذخیره نشدن فایل‌ها در هنگام قطعی برق، هنگ کردن سیستم یا دلایل بی‌شمار دیگر داشته‌اید. به‌عنوان راه حل هم شاید روش‌هایی مانند کپی‌گرفتن از کل فایل‌ها و اطلاعات پروژه در زمان‌های مختلف برای حفظ حالت خاصی از تغییرات را استفاده کرده باشید.

این کار تا حدی جواب می‌دهد، اما در این صورت با انبوهی از دایرکتوری‌هایی که هر لحظه بزرگ و بزرگ‌تر می‌شوند چه می‌کنید؟ و از آن بدتر چگونه آن را با افراد دیگری که با شما در انجام آن همکاری می‌کنند به اشتراک می‌گذارید؟

ابزارهای کنترل نسخه^۱ پاسخی برای این شلختگی‌ها و شلوغی‌هاست. با استفاده از این ابزارها می‌توانید هر لحظه‌ای که مایل بودید تغییرات خود را ثبت کنید، به تغییرات ثبت شده در گذشته برگردید و به راحتی با دوستان و افراد تیمتان روی پروژه‌ای همکاری کنید، بدون اینکه نگران به هم ریختگی و نامنظم‌شدن کدهای پروژه باشید.

روش‌ها و همچنین ابزارهای زیادی برای این کار معرفی شده‌اند و git یکی از این ابزارهاست که هم اکنون به ابزاری کلیدی در عمده‌ی پروژه‌های برنامه‌نویسان تبدیل شده است.

نصب

- نصب بر روی Ubuntu و Debian:

^۱ Version control

```
apt-get install git
```

- نصب بر روی Fedora:

```
yum install git
```

- نصب بر روی ویندوز:

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

پیکربندی

بعد از نصب گیت لازم است تنظیماتی را انجام دهید و همچنین خودتان را به گیت معرفی کنید. این کار به خاطر ثبت تغییرات ایجادشده توسط شما، به نام شما و همچنین شخصی سازی کردن ویژگی های گیت است.

```
git config --global user.name "نام شما"
```

```
git config --global user.email "آدرس ایمیل شما"
```

شروع یک پروژه با گیت

در سیستم های کنترل نسخه به دایرکتوری که کدهای پروژه را در آن نگهداری می کنید مخزن^۲ می گویند. شما در سیستم خود یک مخزن محلی دارید و با ساخت یک پروژه در سرور گیت یک مخزن سمت سرور^۳ خواهید داشت. در یک مخزن گیت محلی^۴، فایل ها در یکی از سه وضعیت اصلی زیر قرار دارند:

- Modified
- Staged

^۲ Repository

^۳ Remote Repository

^۴ Local Repository

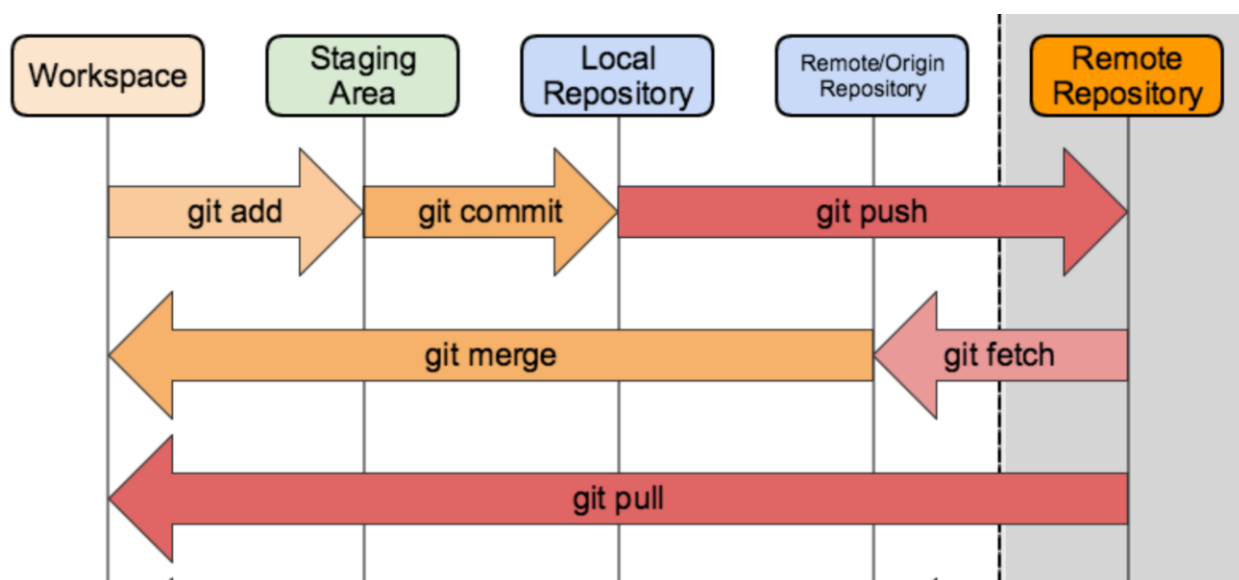
- Committed

وقتی روی فایل‌ها در دایرکتوری local کار می‌کنید و آن‌ها را تغییر می‌دهید فایل‌ها در وضعیت modified قرار می‌گیرند. وقتی آن‌ها را به گیت اضافه می‌کنید به وضعیت staged می‌روند و وقتی آن‌ها را ثبت می‌کنید وضعیت آن‌ها به committed تغییر پیدا می‌کند. (شکل ۱)

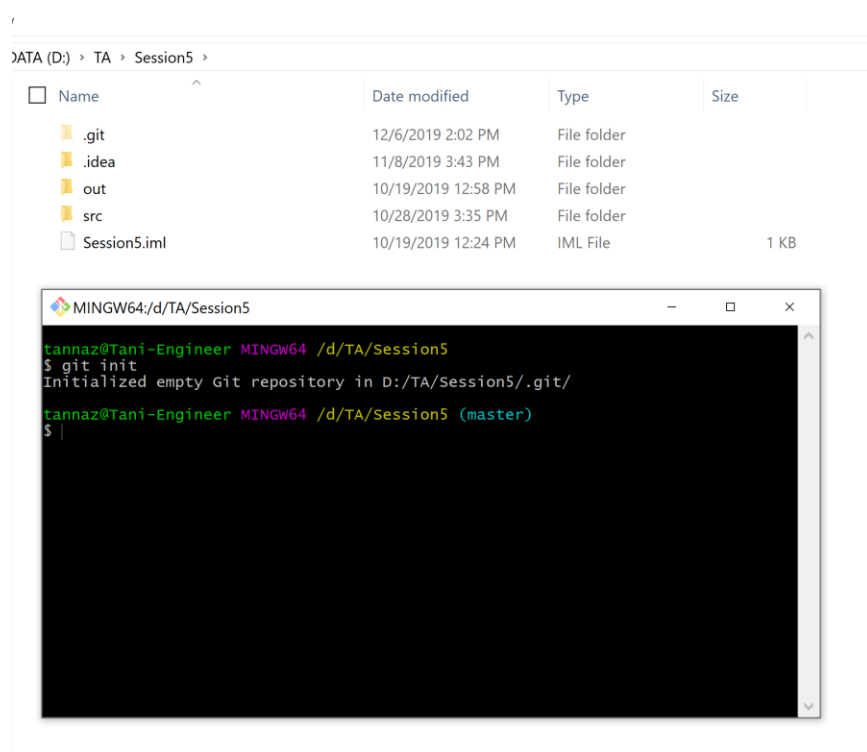
برای ساخت یک مخزن local از دستور زیر در دایرکتوری مربوطه استفاده کنید (شکل ۲).

git init

پس از اجرای دستور یک فولدر با نام git. در دایرکتوری اضافه می‌شود.



شکل ۱: وضعیت فایل‌ها



شکل ۲: ساخت local repository

برای ساخت یک پروژه در سرور گیت مراحل زیر را دنبال کنید.

۱. به لینک گیت دانشکده بروید: <https://git.ceit.aut.ac.ir>
۲. سمت راست بالای صفحه گزینه new project را انتخاب کنید.
۳. برای پروژه خود نام مناسب انتخاب کنید، توضیحات مربوطه را بنویسید و سطح دسترسی آن را با توجه به پروژه خود انتخاب کنید (شکل ۳).
۴. تیک initialize را در صورتی که می‌خواهید یک مخزن از ابتدا سمت سرور بسازید بزنید ولی از آن جایی که در حال حاضر می‌خواهیم فایل‌های یک مخزن محلی را به سرور بفرستیم آن را انتخاب نمی‌کنیم. توجه داشته باشید که فایل README.md برای پروژه های گیت بسیار مهم است پس حتما آن را در نظر بگیرید و توضیحات مربوط به کد را درون آن قرار دهید.

۵. ممکن است در دایرکتوری پروژه فایل‌هایی داشته باشید، که نخواهید گیت آن‌ها را در Status‌ها نشان دهد، و همچنین نخواهید در مخزن اصلی اضافه شوند. برای این کار باید در دایرکتوری پروژه یک فایل به نام `.gitignore` بسازید و در آن، لیست فایل‌ها و دایرکتوری‌هایی را که گیت باید نادیده بگیرد را بنویسید. توجه کنید که خود فایل `.gitignore` باید توسط دستور `add` به پروژه اضافه و `commit` شود. در این فایل، خطوطی که با `#` شروع می‌شوند، به عنوان توضیحات (کامنت) در نظر گرفته می‌شوند (شکل ۴).

۶. پس از ساخت پروژه باید ارتباط بین مخزن `local` و سرور را برقرار کنیم. لینک پروژه را می‌توانید از بالای صفحه کپی کنید (شکل ۵).

لینک پروژه `git remote add origin`

The screenshot shows the GitHub 'Create new repository' interface. It has three tabs: 'Blank project', 'Create from template', and 'Import project'. The 'Blank project' tab is active. The form includes the following fields and options:

- Project name:** A text input field containing 'Shapes'.
- Project URL:** A text input field containing 'http://git.ceit.aut.ac.ir/95131122/'.
- Project slug:** A text input field containing 'shapes'.
- Create a group:** A link with a checkbox to 'Want to house several dependent projects under the same namespace? Create a group.'
- Project description (optional):** A text area containing 'This project is for practicing inheritance in java'.
- Visibility Level:** Three radio buttons: 'Private' (selected), 'Internal', and 'Public'. Descriptions are provided for each.
- Initialize repository with a README:** A checkbox that is currently unchecked.
- Buttons:** A green 'Create project' button and a 'Cancel' button.

شکل ۳: ساخت پروژه

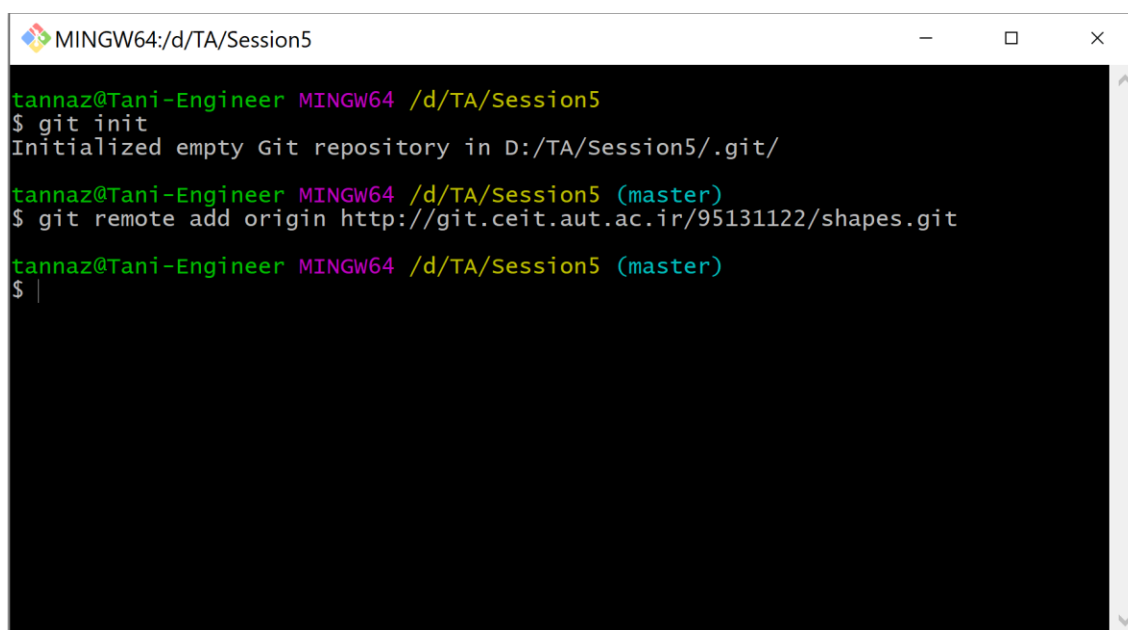
S **Shapes** Private Add license

This project is for practicing inheritance in java.

Project ID: 640

0 ☆ Star HTTP http://git.ceit.aut.ac.ir + Global

شکل ۴: لینک پروژه



```

MINGW64:/d/TA/Session5
tannaz@Tani-Engineer MINGW64 /d/TA/Session5
$ git init
Initialized empty Git repository in D:/TA/Session5/.git/
tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$ git remote add origin http://git.ceit.aut.ac.ir/95131122/shapes.git
tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$ |
  
```

شکل ۵: اتصال به سرور

اضافه کردن فایل‌ها

برای اضافه کردن فایل‌ها به گیت از دستور add استفاده می‌کنیم. به صورت زیر:

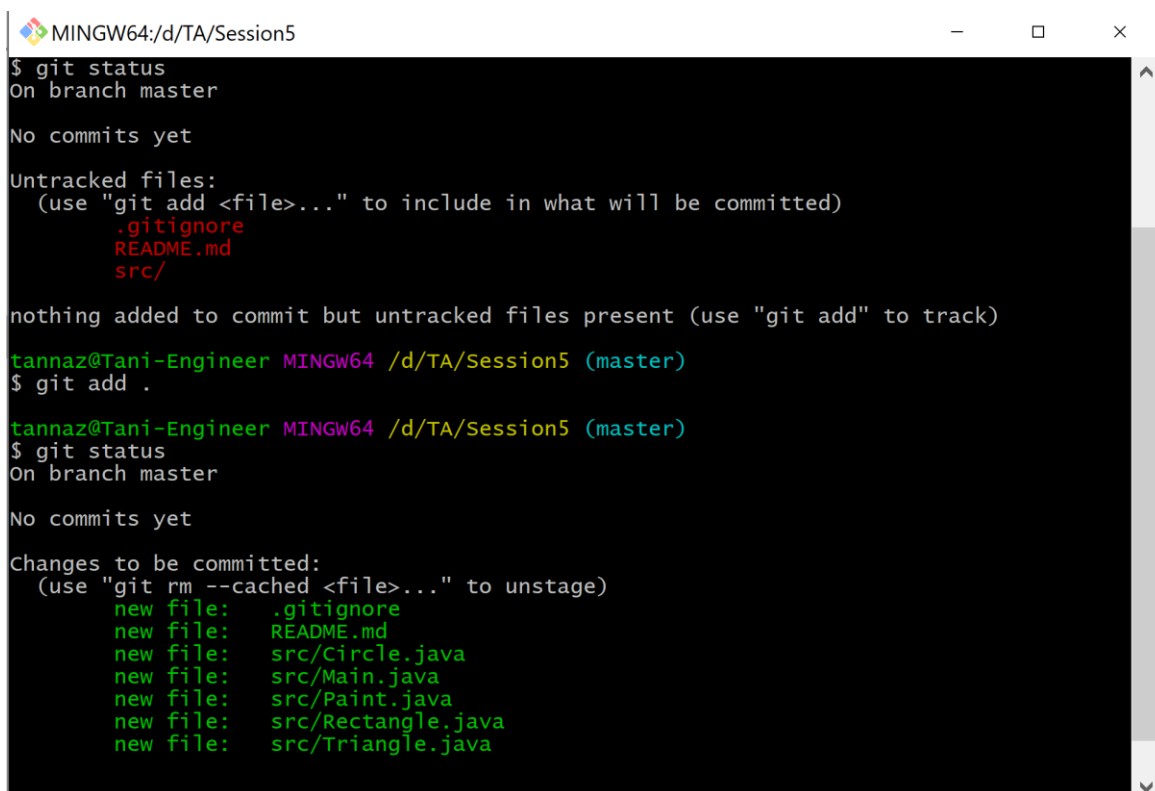
فایل ۱ فایل ۲ فایل ۳ `git add`

و برای افزودن تمام فایل‌های تغییر داده شده دستور را به صورت زیر استفاده می‌کنیم:

`git add` یا `git add -A`.

با استفاده از دستور status می‌توانید وضعیت فایل‌ها را مشاهده کنید. فایل‌ها از وضعیت modified (قرمز رنگ) به وضعیت staged (سبز رنگ) تغییر می‌کنند (شکل ۶).

git status



```
MINGW64:/d/TA/Session5
$ git status
On branch master

No commits yet

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

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

tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$ git add .

tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   README.md
        new file:   src/Circle.java
        new file:   src/Main.java
        new file:   src/Paint.java
        new file:   src/Rectangle.java
        new file:   src/Triangle.java
```

شکل ۶: اضافه کردن فایل‌ها

ثبت تغییرات

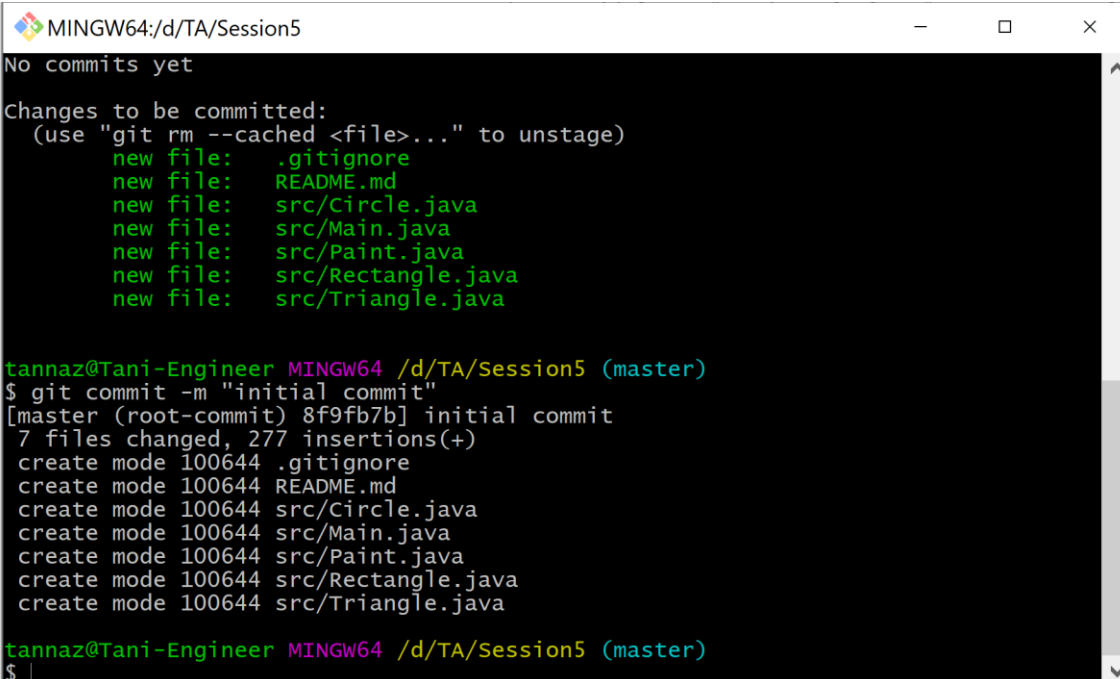
برای ثبت تغییرات یا به اصطلاح کامیت کردن تغییرات، از دستور commit استفاده می‌کنیم. در این مرحله فایل‌هایی را که با دستور add به حالت stage برده‌ایم در سیستم گیت ثبت می‌کنیم. برای ثبت هر تغییر نیاز است یک پیام هم با آن ثبت شود تا معلوم شود در این قسمت از تغییرات لحاظ شده چه کار کرده‌ایم، یا چه تغییراتی داده‌ایم.

برای مثال اگر ما یک فایل متنی برای نوشتن توضیحات پروژه به نام readme.md ساخته باشیم و با دستور add آن را برای کامیت شدن آماده کرده باشیم، می‌توانیم به

همراه کامیت خود یک پیام با مضمون add read me file ثبت کنیم که تغییرات، برای مطالعه در آینده شفافتر باشند.

git commit -m "پیام شما"

قرارداد استاندارد برای پیام کامیت : ۱. در زمان حال نوشته شود ۲. در هنگام استفاده از سویچ -m پیام کمتر از ۵۰ کاراکتر باشد.



```

MINGW64:/d/TA/Session5
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .gitignore
    new file:   README.md
    new file:   src/Circle.java
    new file:   src/Main.java
    new file:   src/Paint.java
    new file:   src/Rectangle.java
    new file:   src/Triangle.java

tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$ git commit -m "initial commit"
[master (root-commit) 8f9fb7b] initial commit
 7 files changed, 277 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 src/Circle.java
 create mode 100644 src/Main.java
 create mode 100644 src/Paint.java
 create mode 100644 src/Rectangle.java
 create mode 100644 src/Triangle.java

tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$
  
```

شکل ۷: ثبت تغییرات

ارسال کد به مخزن ریموت

برای افزودن تغییرات کامیت شده در مخزن local به مخزن remote از دستور push استفاده می‌کنیم به صورت زیر:

نام شاخه origin push -u git

git push -u origin master


```
tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$ git push -u origin master
warning: redirecting to https://git.ceit.aut.ac.ir/95131122/shapes.git/
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 2.40 KiB | 491.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0)
To http://git.ceit.aut.ac.ir/95131122/shapes.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$
```

شکل ۸: ثبت تغییرات

دریافت آخرین تغییرات از مخزن ریموت

برای دریافت تغییرات کامیت شده به مخزن ریموت از دستور pull استفاده می‌کنیم:

`git pull`

شاخه‌ها

برنج‌ها شاخه‌های مختلفی را برای توسعه ایجاد می‌کنند. فرض کنید که در حال توسعه یک اپلیکیشن هستید و قصد دارید نسخه‌ی آینده اپلیکیشن خود را همزمان با نسخه‌ی فعلی توسعه دهید. اضافه کردن تمام این تغییرات با هم ممکن است باعث شلوغی و بی‌نظمی روند توسعه، و همچنین تداخل فایل‌های هم‌نام شود. در سناریویی دیگر، شما و همکاران همزمان در حال توسعه‌ی یک اپلیکیشن هستید. هرکدام یک شاخه برای خود ایجاد می‌کنید و بدون این که در کدهای هم تداخل ایجاد کنید، برنامه را توسعه می‌دهید و در آخر کدها را تلفیق می‌کنید. با استفاده از برنج‌ها در گیت می‌توانید یک مسیر جدید برای توسعه هر ویژگی ایجاد کنید و همچنین می‌توانید در پایان ویژگی‌های کامل شده را به برنج اصلی اضافه کنید. در گیت به طور پیش فرض، برنج اصلی به نام master است.

برای ساختن یک برنج توسعه‌ی جدید، از دستور زیر استفاده کنید:

نام شاخه git branch

برای نمایش لیستی از برنچ‌ها از دستور زیر استفاده کنید:

git branch -va

برای سوویچ کردن به شاخه دیگر از دستور زیر استفاده کنید:

نام برنچ git checkout

```
tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$ git branch developer1

tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (master)
$ git checkout developer1
Switched to branch 'developer1'

tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (developer1)
$ git branch -va
* developer1          8f9fb7b initial commit
master               8f9fb7b initial commit
remotes/origin/master 8f9fb7b initial commit

tannaz@Tani-Engineer MINGW64 /d/TA/Session5 (developer1)
$
```

شکل ۹: شاخه‌ها

تلفیق کدها

در بخش قبل نحوه‌ی استفاده از برنچ‌ها گفته شد. حال اگر بخواهیم این شاخه‌های جدا را دوباره با شاخه‌ی اصلی تلفیق کنیم، باید از دستور merge استفاده کنیم:

شاخه git merge

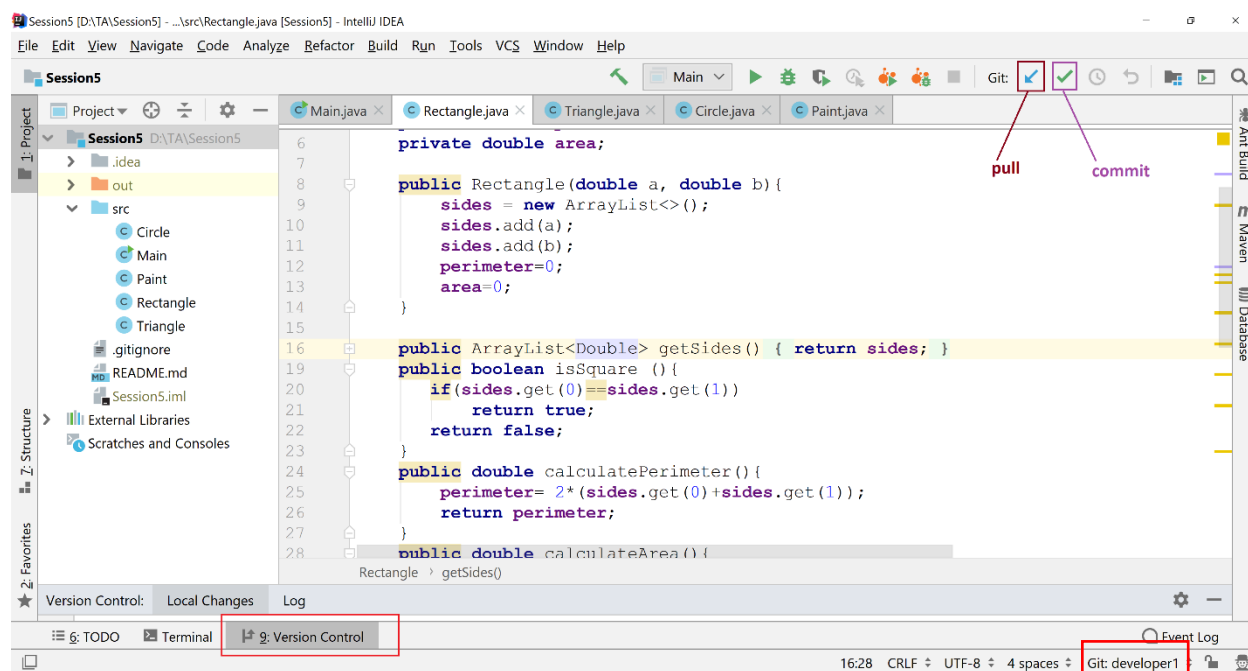
با این دستور محتویات شاخه‌ی داده شده به دستور، با شاخه‌ی فعلی ترکیب می‌شود. اگر کدها با هم تداخل داشته باشند عملیات merge به صورت خودکار انجام نمی‌شود. کدهای دارای تفاوت در یک ویرایشگر باز می‌شوند. باید ابتدا تداخل کدها را برطرف

کرده و سپس merge کنید (برای تسهیل این فرایند می‌توانید از امکانات محیط برنامه نویسی IntelliJ برای تلفیق و برطرف کردن تداخل استفاده کنید).

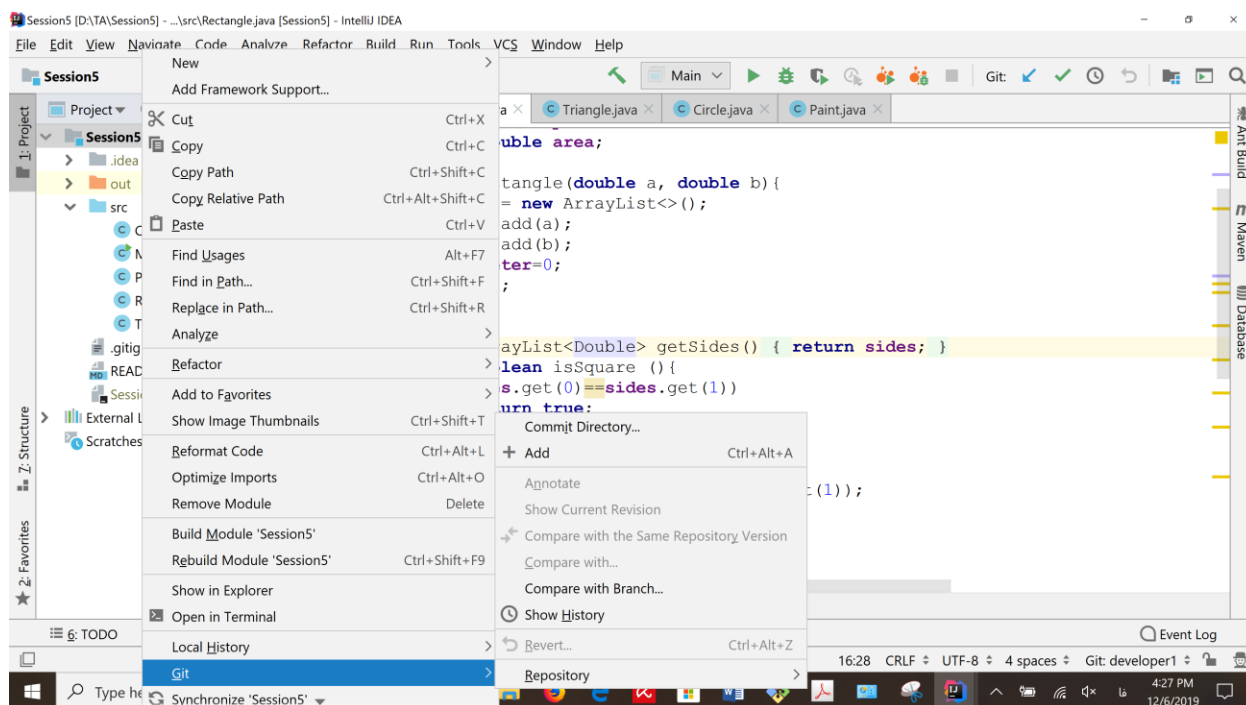
کار با گیت از طریق محیط برنامه نویسی IntelliJ

تمام دستوراتی که در بالا گفته شد از طریق محیط برنامه نویسی IntelliJ هم فراهم می‌شود که در تصاویر زیر مشاهده می‌کنید. با راست‌کلیک روی پروژه و انتخاب گزینه git می‌توانید عملیات مربوطه را مشاهده کنید.

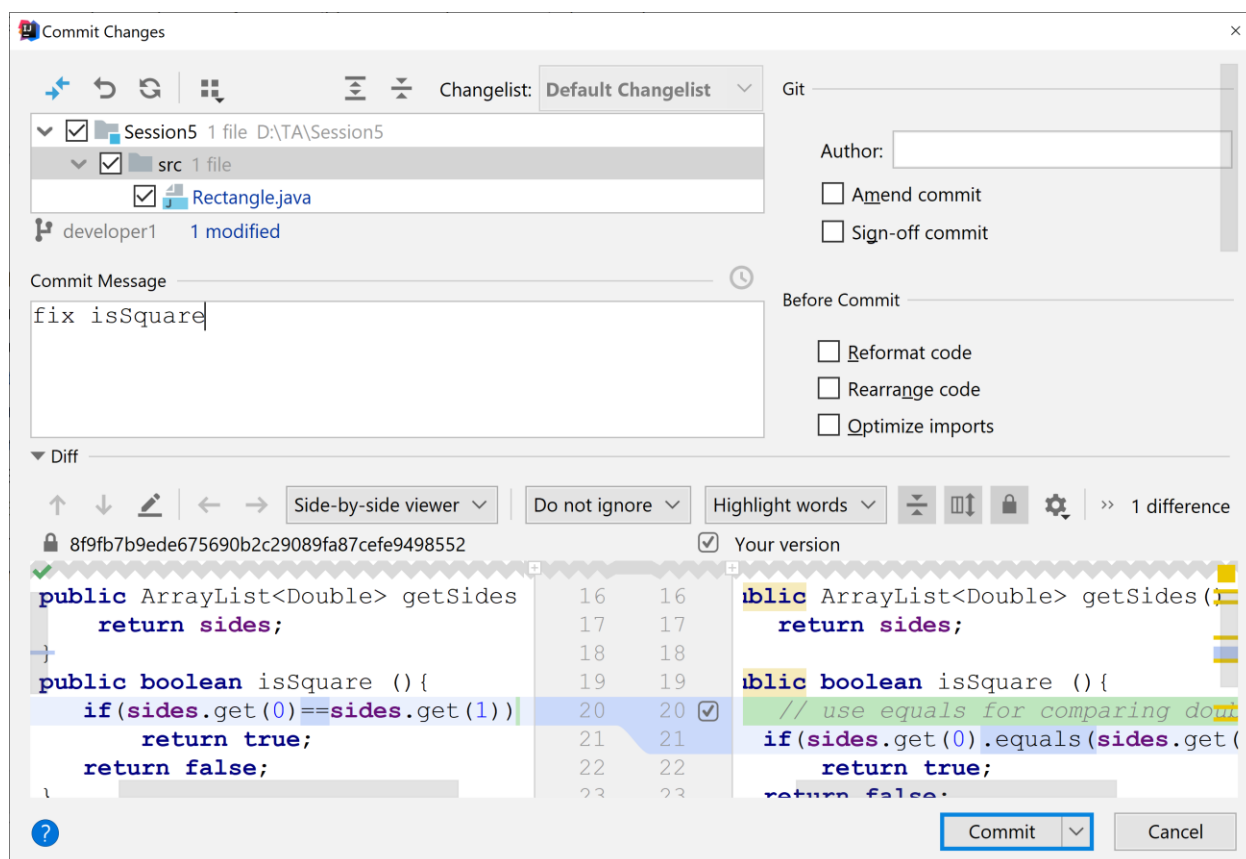
پایین صفحه سمت راست هم نام شاخه را می‌بینید که می‌توانید همینجا بین شاخه‌ها جابجا شوید.



شکل ۱۰: دسترسی سریع به گیت



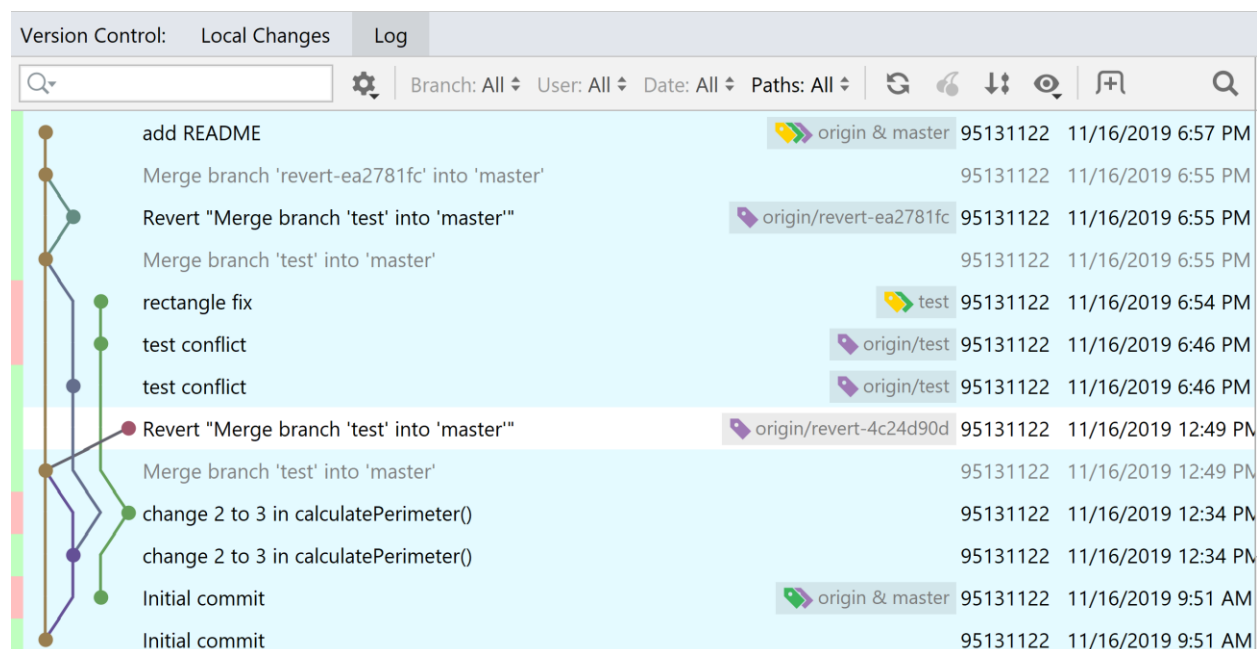
شکل ۱۱: دسترسی سریع به گیت-۲



شکل ۱۲: کامیت از طریق IntelliJ

همان طور که در شکل ۱۲ می‌بینید در هنگام کامیت، فایل‌های تغییر داده شده با جزئیات نمایش داده می‌شوند.

از قسمت version control هم می‌توانید گزارش کامل تغییرات فایل‌ها را مشاهده کنید (شکل ۱۳).



شکل ۱۳: مشاهده گزارش تغییرات