**به نام خداوند بخشنده مهربان**

**محمدرضا مسیب زاده**

تمرین جاوا 8:

تمرین i :

اول از همه مفهموم استاتیک بودن رو بفهمیم:

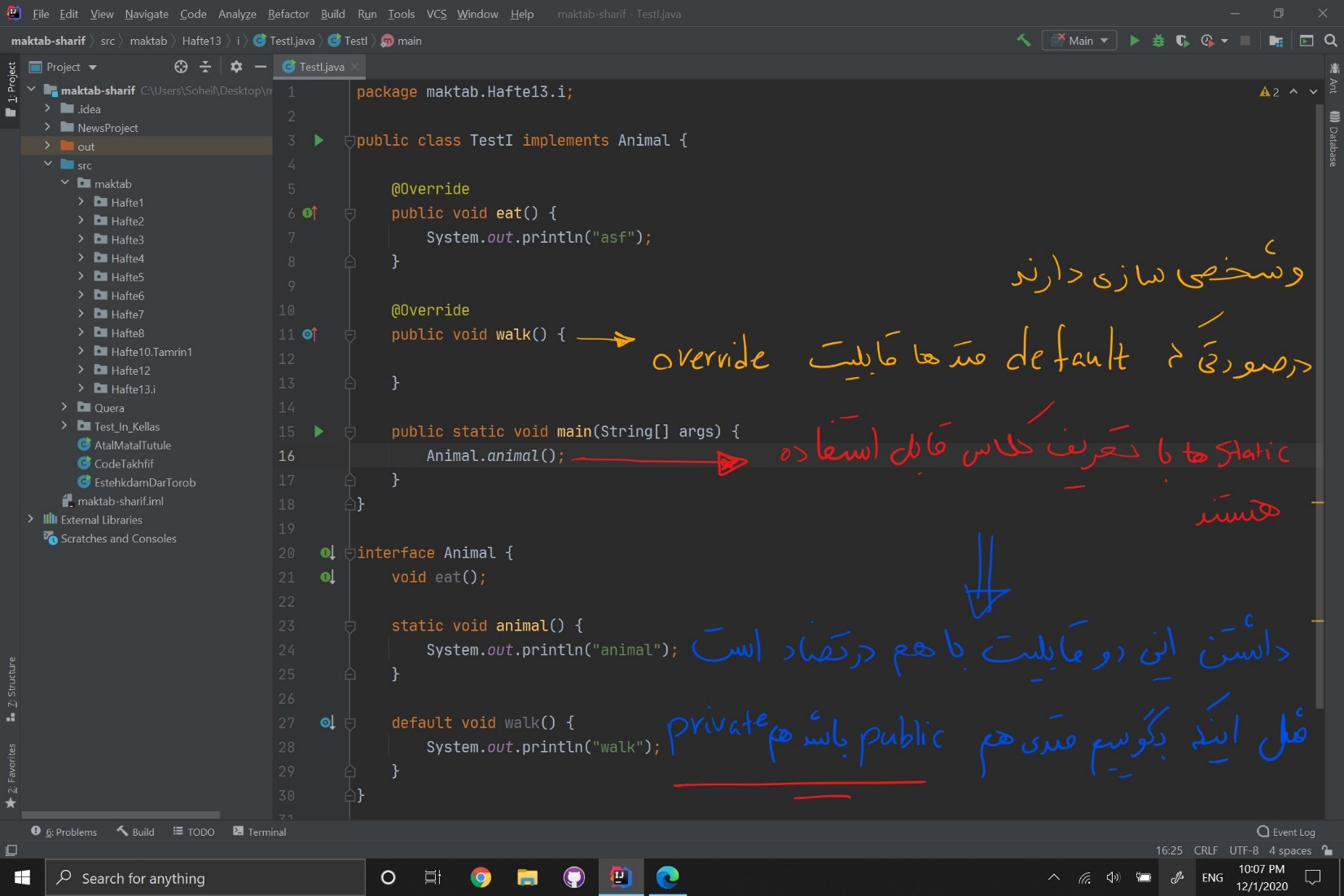
متدها یا متغییر های استاتیک همزمان با ایجاد شدن کلاس توسط JVM تعریف میشوند پس بعد از تعریف کلاس متد استاتیک با آن تعریف میشود و حتی بعد از ارث بری قابل override کردن نیست چون جزئی از تعریف کلاس ماست

در کل ما سه نوع متد میتوانیم در اینترفیس ها تعریف کنیم:

1. متدهای بدون بدنه که صرفا برای این هستن که کلاس های مختلف نیاز به پیاده سازی های مختلفی برای آن دارند پس این متد فقط تعریفی انتزاعی است برای کلاس هایی که ان را به دلخواه خود تعریف میکنند
2. متد های default ، این متد ها تعریف و بدنه ای پیش فرض از متد را به ما میدهند اما کلاس به دلخواه خود میتواند آن ها را شخصی سازی کرده یا نکند
3. متد های کلاسی ،تعریف واضحی از اینترفیس هستند که با پیاده سازی اینترفیس آن متد هم برای ما وجود دارد و قابل تغییر نیست چون جز جدانشدنی اینترفیس ماست.

حال وقتی ما میگوییم متدی هم استاتیک باشد هم default ،از یک طرف استاتیک است و کلاسی که اینترفیس ما را پیاده سازی میکند باید آن را بپذیرد و فقط توانایی استفاده از آن متد را با آوردن نام اینترفیس قبل از آن دارد از طرفی میگوییم default است و کلاس ما میتواند آن را به دلخواه خود تغییر و شخصی سازی کند.

این دو مقوله با هم در تضاد هستند ، همانطور که نمیتوانیم بگوییم متدی هم public است و هم private یا متغییری هم استرینگ است و هم اینتجر.



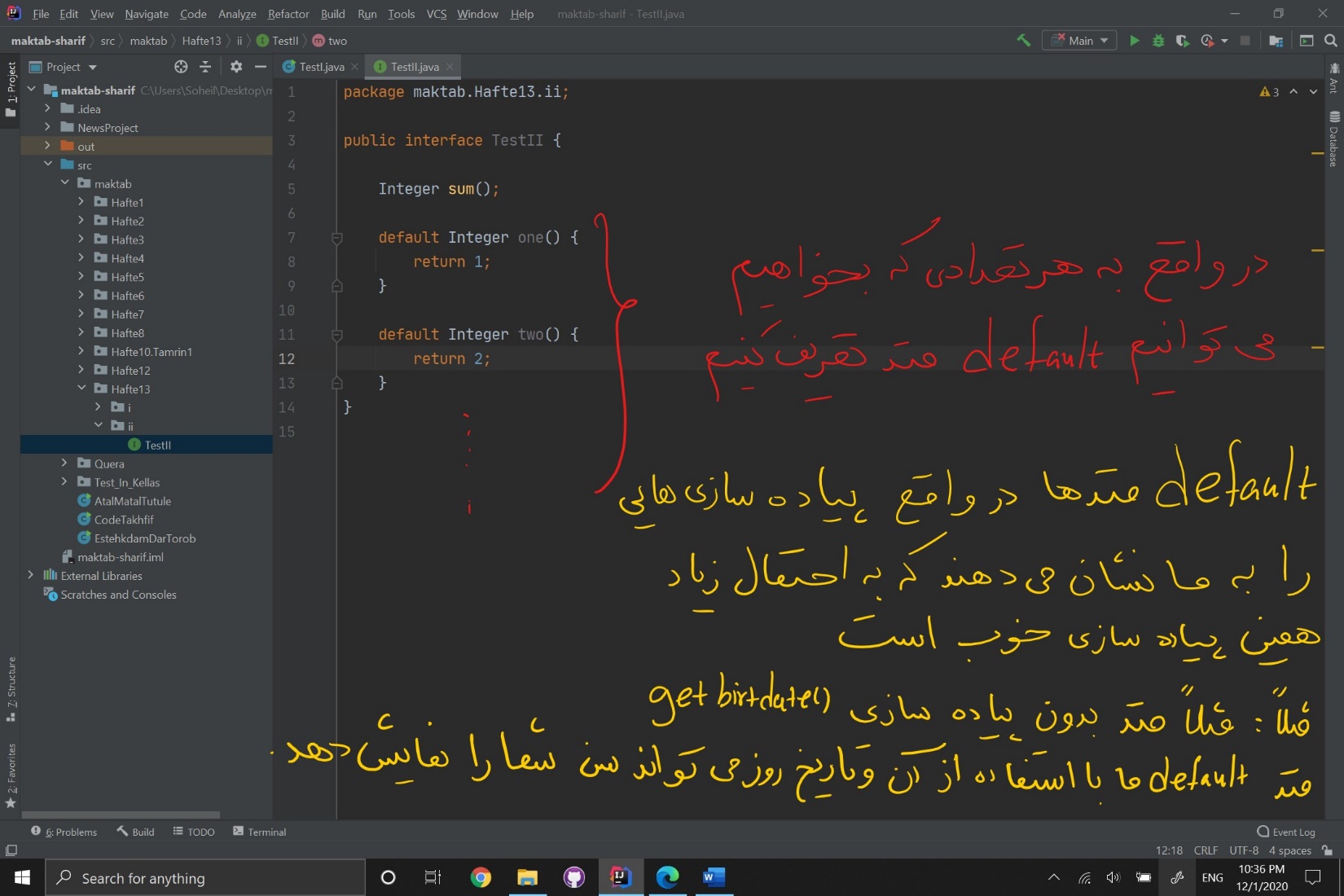
تمرین ii :

ما به هر تعداد که بخواهیم میتوانیم متد default داشته باشیم

در واقع متد های defualt کاربردی که دارن برای این است که متدهای دیگری که پیاده سازی نشده اند را در صورتی که توسط ما پیاده سازی شوند میتوانند اطلاعات خوبی به ما بدهند که دیگر نیازی نیست این متدها را به صورت تعریف نشده بگیریم و خودمان از صفر تعریف کنیم در صورت دلخواه میتوانیم آن ها را override کنیم

مثلا متد پیاده سازی نشده getBirthdate() قرار است توسط ما پیاده سازی شود و تاریخ تولد را برگرداند

خب به احتمال زیاد میتوان گفت میتوانیم متد defualt داشته باشیم به اسم getAge() که با استفاده از متد getBirthdate() سن را به ما نمایش دهد



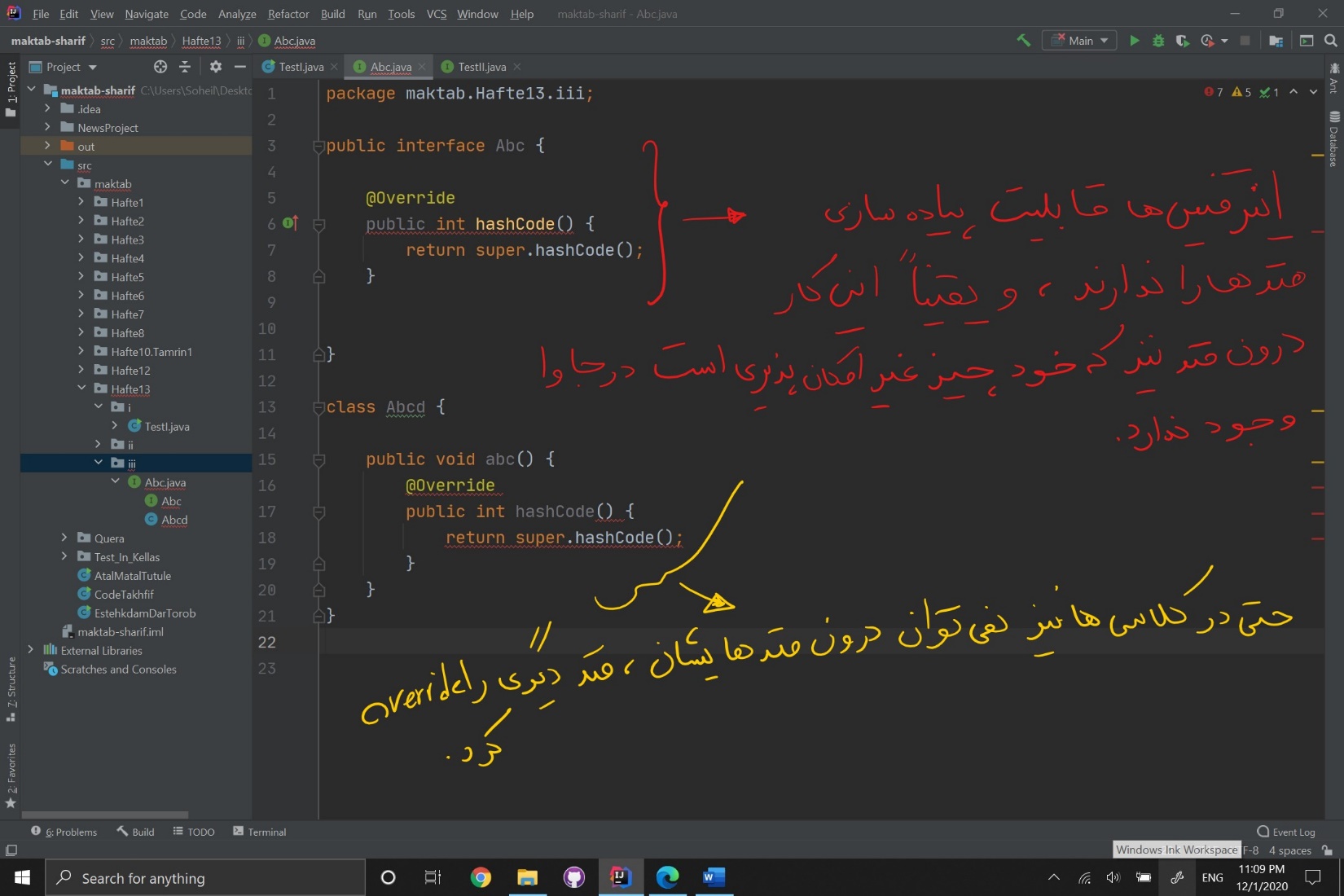
تمرین iii :

به نظرم این سوال خودش دو بخشه

این که اول متدها اصلا درون خودشون قابلیت override کردن متد کلاس دیگیری رو دارن؟!

دوم اینکه ایا اصلا درون اینترفیس ها که متدهای default وجود دارن ما میتونیم چیزی رو override کنیم؟!

هردو سوال جوابی منفی دارن، در واقع ما درون هیچ متدی نمیتونیم متدی دیگر رو override کنیم و همچنین درون اینترفیس ها که خودشون محل به وجود اوردن چیزهای بدون پیاده سازی هستند نمیتونیم متدی رو override کنیم

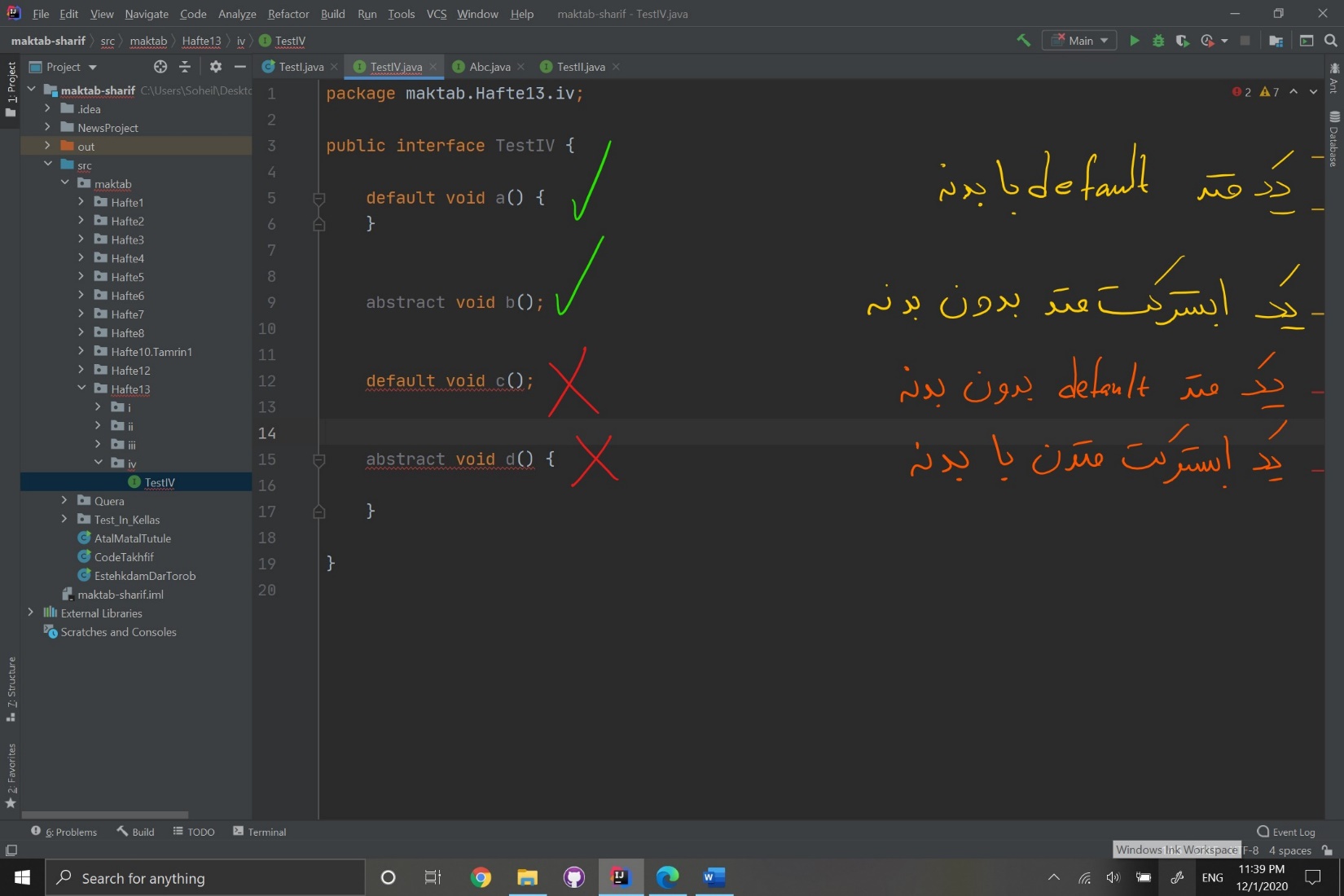


تمرین iv :

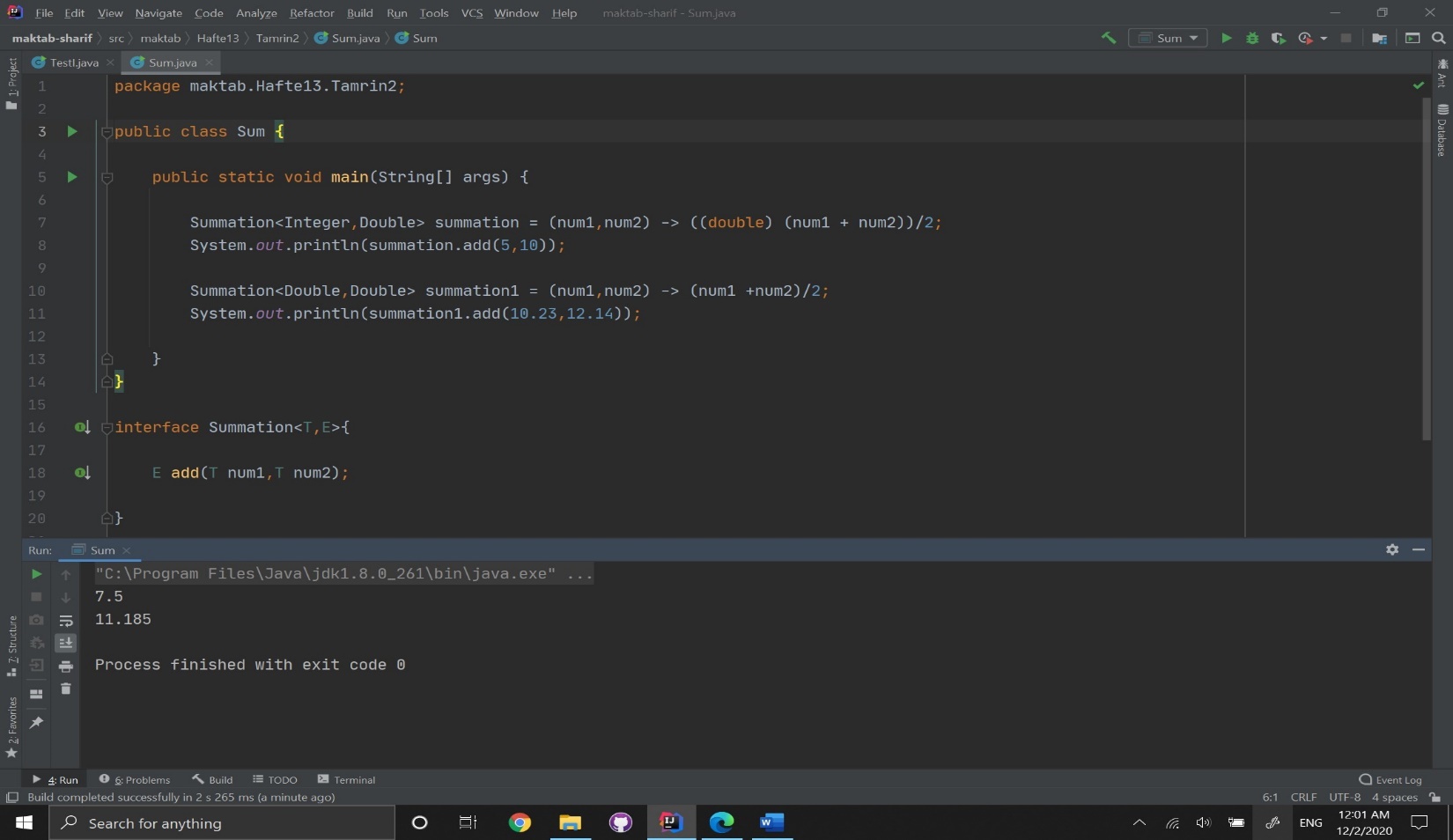
این که بگوییم متدی هم ابسترکت باشد هم default باهم در تناقض است

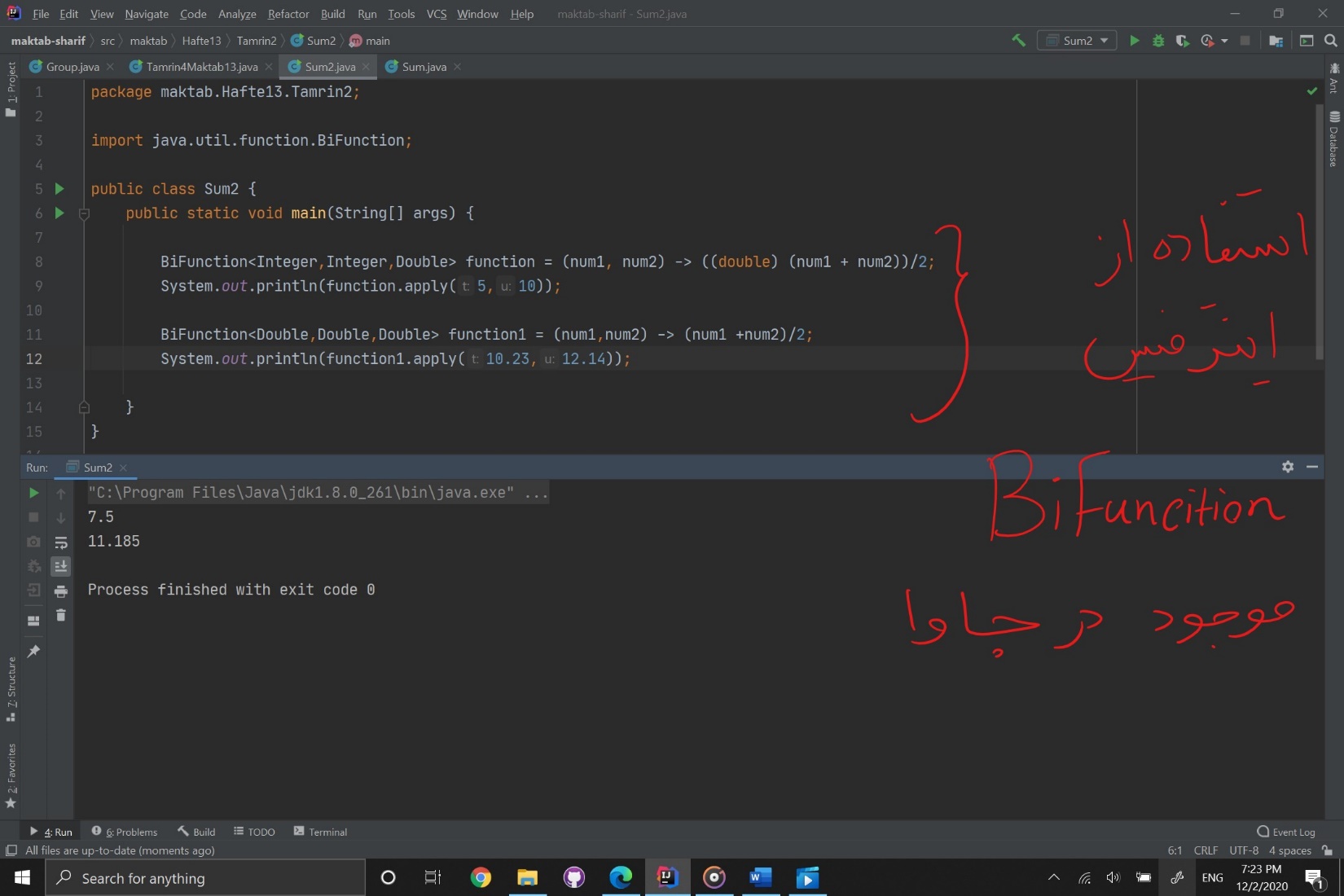
متد abstract ، متدی است که بدنه ای ندارد در حالی که متد default متدی هست که دارای بدنه است

پس همانطور که متدی هم بدنه داشته باشد و هم بدنه نداشته باشد قابل تصور نیست ، به همان صورت متدی که هم default باشد و هم abstract در جاوا امکان پذیر نیست.

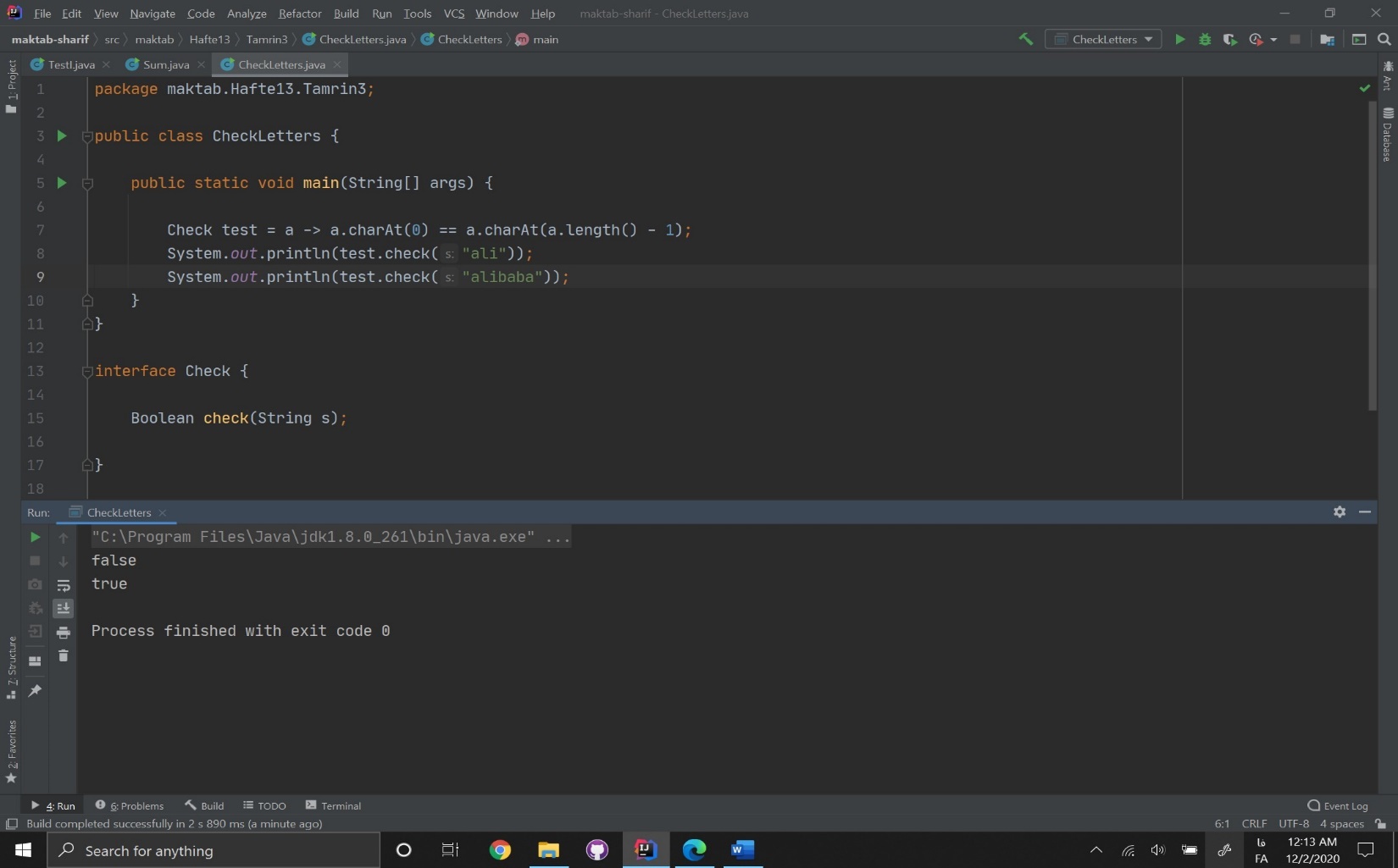


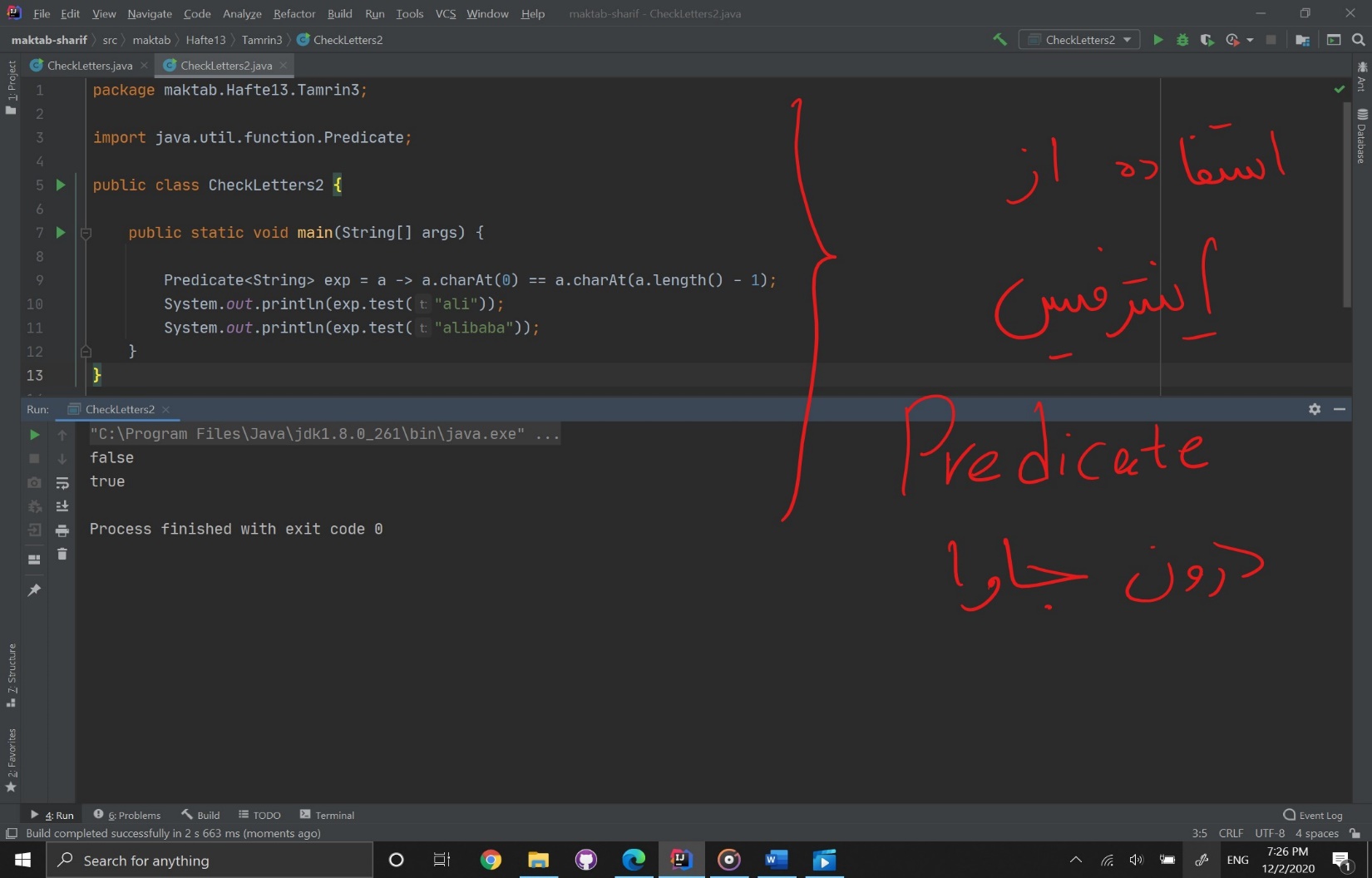
تمرین 2 :



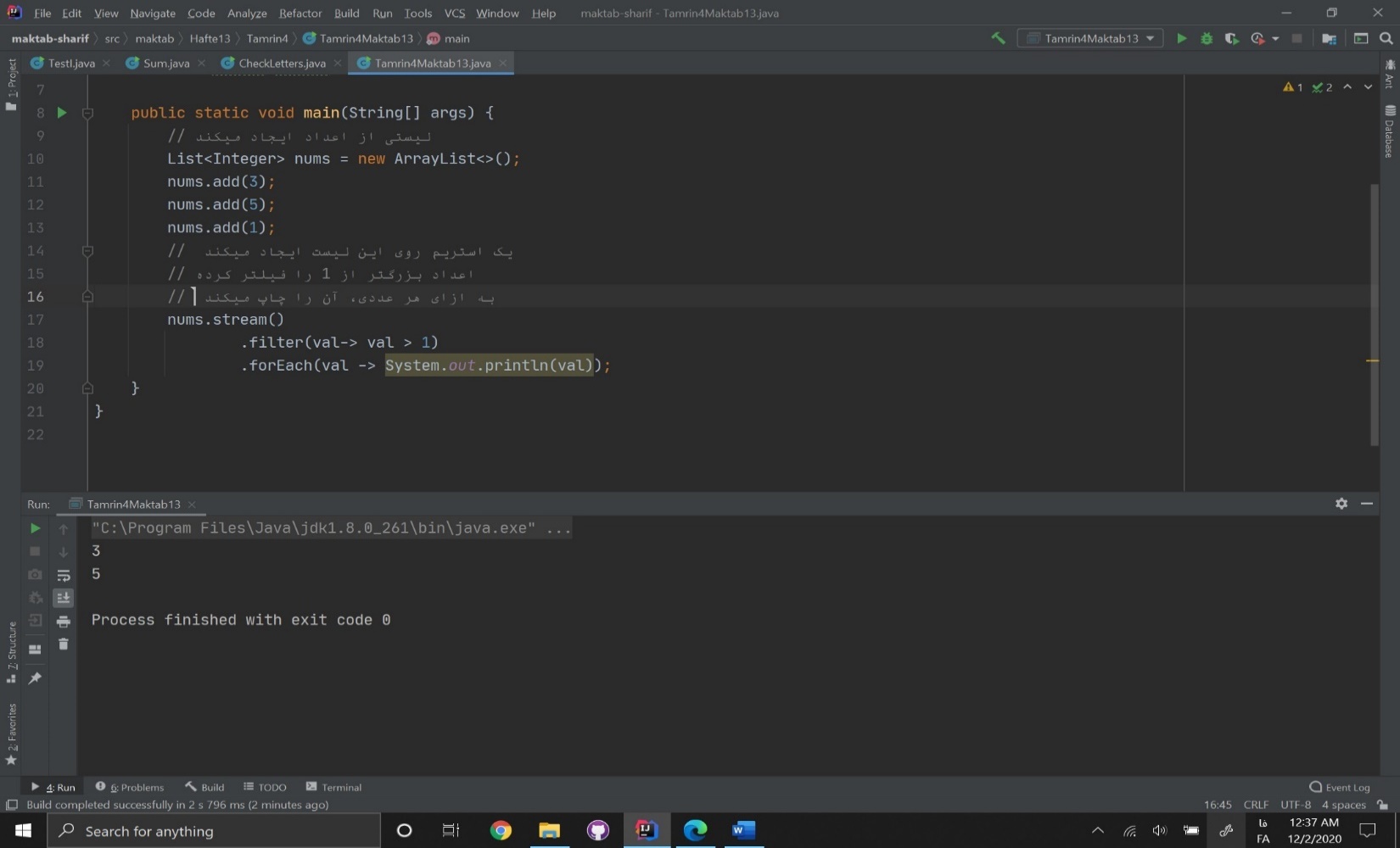


تمرین 3 :

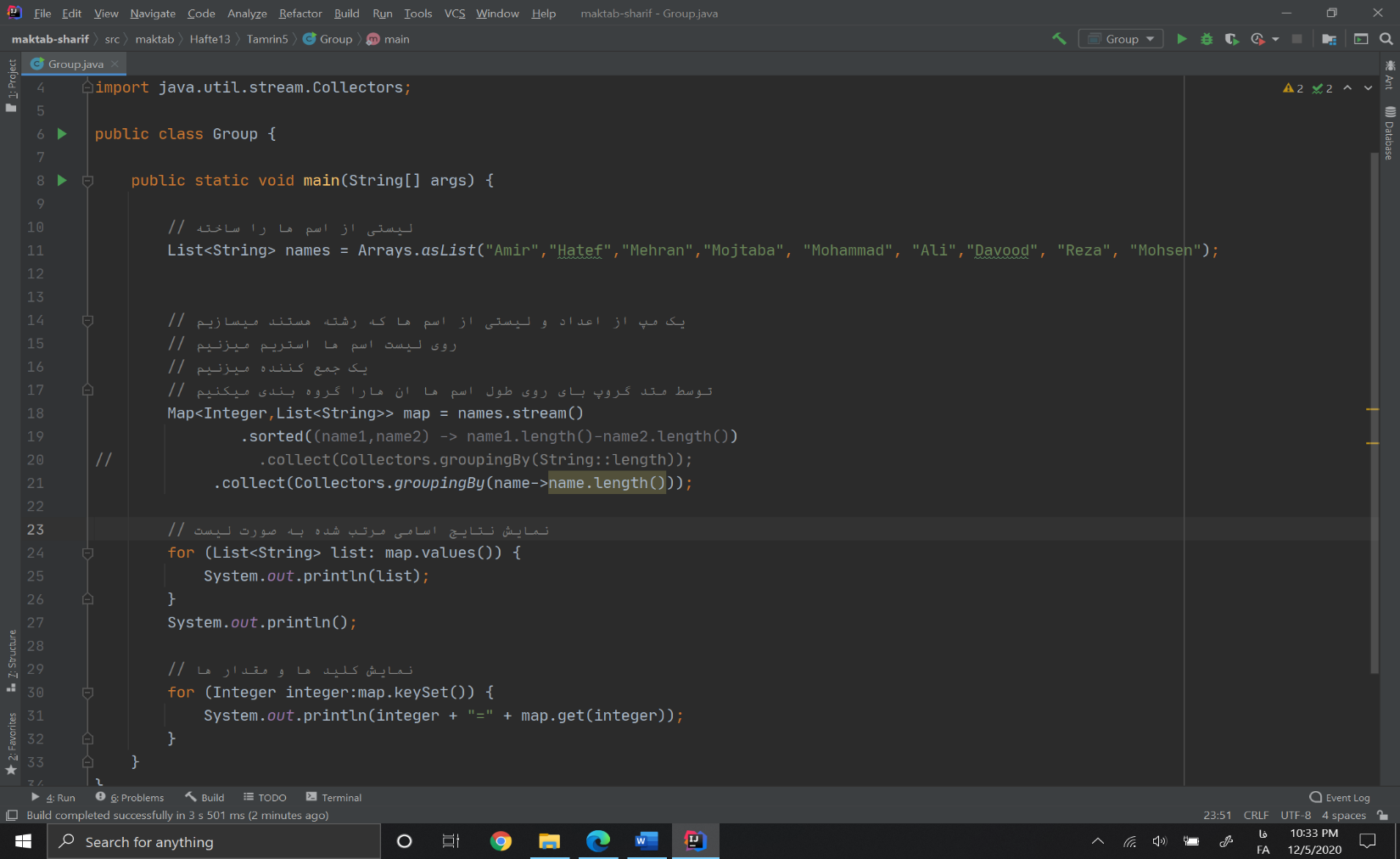




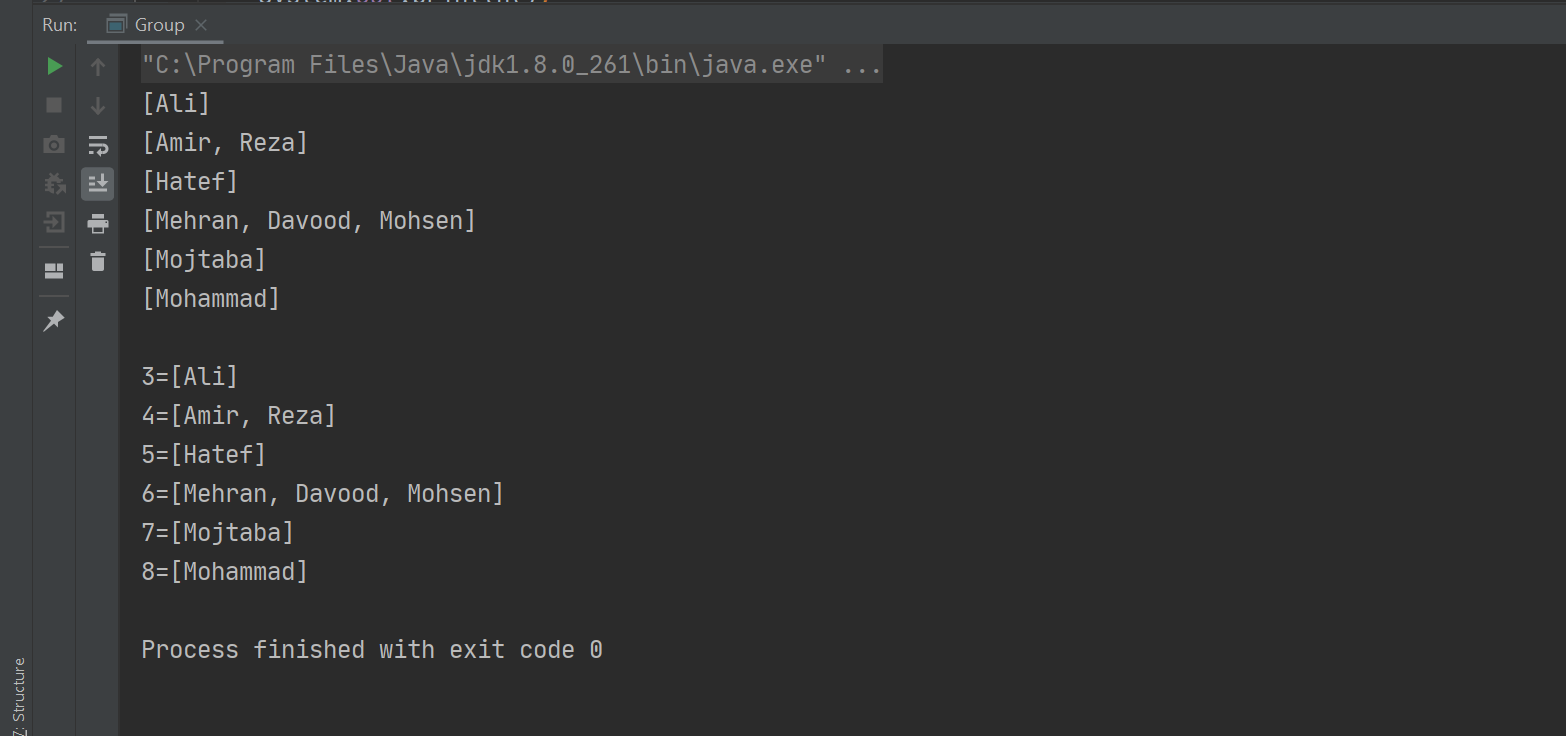
تمرین 4 :



تمرین شماره 5 :



اجرای تمرین 5:



تمرین پایگاه داده:

تمرین1 :

<https://github.com/M-Mosaiebzadeh/Maktab_W13.git>

تمرین 2: حمله به پایگاه داده

2-1-sql injection را توضیح دهید.

یکی از انواع حملات به صفحات وب است که هکرها از این روش میتوانند به بانک اطلاعاتی سایت شما دست یابند

به این وسیله کوئری های مخرب به پایگاه داده تزریق کرده و باعث بروز نقص های امنیتی شدید میشوند

با اجرای موفقیت آمیز دستورات مخرب , اطلاعات و داده های مهم سایت افشا خواهند شد

در این روش هکرها با استفاده از فرم تماس , فرم ارسال نظر , صفحه ورودی , بخش جستجو , صفحه پرداخت و …. اقدام به ارسال کوئری های مخرب میکند. در نتیجه کدها پس از ارسال توسط هکر به عنوان یک دستور در سایت اجرا میشوند.

پس از اجرای کوئری های مخرب , هکر میتواند به اطلاعات مهمی از جمله نام های کاربری , رمز های عبور , شماره حساب های وارد شده در سایت و … دست یابد.

مثال:

SELECT UserID FROM Users WHERE UserName='[user]’ AND Password='[pass]’

برای مثال فرض کنید– –; myuserرا در فیلد نام کاربری و wrongpass را در فیلد رمزعبور وارد کرده باشیم. در اینصورت کوئری زیر را خواهیم داشت:

SELECT UserID FROM Users WHERE UserName='myuser';-- AND Password='wrongpass'

با اینکار پسورد غیر فعال شده و کاربر با یوزر نیم myuser میتواند وارد دیتابیس شود

روش های مقابله با حملات SQL

1. یکی از مهم ترین روش های مقابله با حملات SQL بررسی دائمی اطلاعات ورودی به سایت است. به دلیل ارسال کدهای مخرب از طریق فرم ها , باید امنیت فرم ها را به طوری افزایش داد که از ورود هرگونه Query های غیرمجاز جلوگیری شود
2. دقت داشته باشید هنگامی که پیام و یا هشداری بر روی سایت قرار میدهید , این پیام یا هشدار شامل نکته هایی جهت کمک به هکرها برای شناسایی ضعف سایت نباشد.
3. بررسی مکرر دسترسی کاربران به بانک های اطلاعاتی و دیتابیس ، تا درصورت افشا شدن اطلاعات یک کاربر دیگر امکان دسترسی به بانک اطلاعاتی سایت وجود نداشته باشد.
4. از رمز های عبور قوی در بانک های اطلاعاتی خود استفاده کنید تا هکرها نتوانند به آسانی آنها را شناسایی کنند.
5. جهت جلوگیری از ورود هکر ها به سایت خود از WAF فایروال اپلیکیشن وب استفاده کنید.

2-2- چرا فایروال ویندوز نمیتواند جلوی این حملات را بگیرد و چند فایروال مناسی برای جلوگیری از حملات sql injection بیان کنید.

فایروال ویندوز یا به طور کلی فایروال های سیستم عامل ها به فایروال های شبکه معروف هستن این فایروال ها در لایه 3و 4 لایه بندی OSI فعالیت میکنند در صورتی که حملات sql injection در لایه 7 مدل OSI یعنی لایه اپلیکیشن اتفاق میفتد

برای جلوگیری ازاین حملات ما فایروالی نیاز داریم که به جای کار روی لایه 4 و پروتکل های TCP/IP روی لایه 7 یعنی لایه اپلیکیشن و روی پروتکل های HTTP/FTP/HTTPS و .. کار کند

برای همین باید از WAF ها یا همان Web Application Firewall ها استفاده کرد

مزایای فایروال نرم افزار کاربردی وب WAF

امنیت وب سایت را در مقابل حملات SQL injection، تامین می کند.

از وب سایت، در مقابل حملات Cross-site scripting و directory traversal محافظت می کند.

از حملات دستکاری پارامتر، کوکی و نشست جلوگیری می کند.

با بررسی گواهی SSL ترافیک های encrypt شده، از وب سایت، در برابر تهدیدات محافظت می کند.

امنیت وب سایت را در مقایل حملات DDoS تامین می کند.

از XML همراه با اعتبار سنجی SOAP messages و XPath، محافظت می کند.

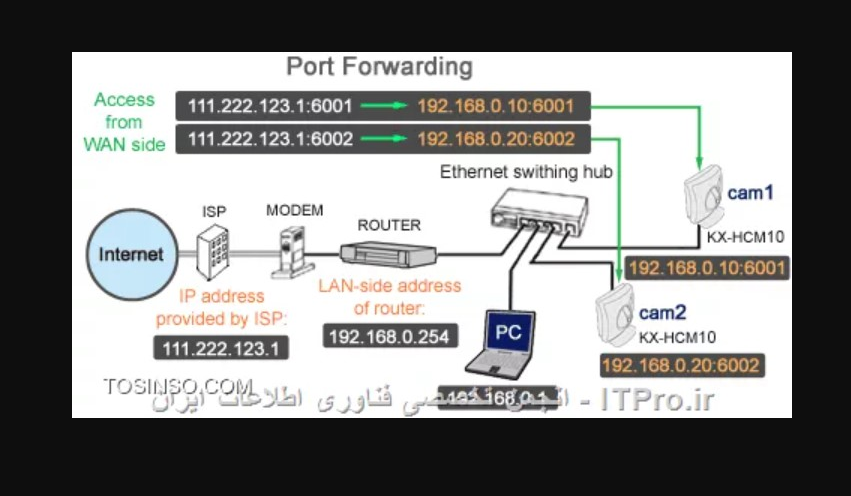
دفاع در مقابل حملات injection را ارائه می دهد و مستندات XML ای که دارای محتوای مخرب می باشند را Block می کند.

تمرین شبکه :

تمرین 1: کارکرد port forwarding را توضیح دهید.

پورت فرواردینگ درخواست های دریافتی از شبکه های خارجی را به مقصد مشخص شده در شبه داخلی هدایت میکند این سرویس میتواند کاربردهای فراوانی در شبکه داشته باشد ولی یکی از بارزترین و پرکاربردترین مصارف این ابزار دسترسی از راه دور به سیستم های کامپیوتری یا همان Remote Desktop میباشد همانطور که میدانید امروزه با سرویس هایی از قبیل Team Viewer و برخی سرویس های رایگان دیگر شما به راحتی امکان اتصال و دسترسی به هر سیستمی در هر جای دنیا به صورت Remote را دارید که با استفاده از این امکانات Port forwarding دیگری نیازی به استفاده از ابزارها و نرم افزارهای جانبی نخواهد بود .

بطوریکه با انجام برخی تنظیمات بر روی روتر خانگی خود و فوروارد کردن پورت 3389 که همان پورت Remote desktop Protocol میباشد به مقصد ای پی کامپیوتری که قصد اتصال از راه دور به ان را دارید به راحتی میتوانید هر زمان و از هر مکانی به کامپیوتر خانگی خود متصل شوید .همانطوری که در تصویر زیر نیز مشاهده میکنید تمامی درخواست های دریافتی از شبکه خارجی WAN دقیقا به وسیله ای که پورت به ای پی ان وسیله فوروارد شده هدایت میگردد.یکی از مصارفی که امروزه در بسیاری از اماکن میتوان بعنوان مثال برای این سرویس نام برد در مورد دوربین های مدار بسته میباشد که شما از طریق راه دور میتوانید به شبکه محلی دوربین های خود وصل و دوربین مد نظر خود را تماشا کنید



از کاربردهای عمده دیگری که برای این سرویس میتوان نام برد عبارتند از :

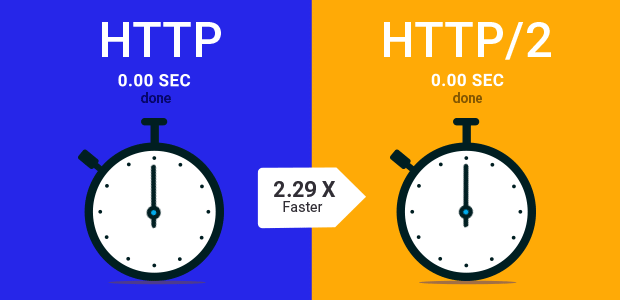
* دسترسی امن به شبکه محلی از راه دور توسط خط دستور بر روی شبکه اینترنت
* دسترسی امن به شبکه محلی توسط پروتکل FTP برای نقل و انتقال فایل
* دسترسی به شبکه های گیم محلی از طریق اینترنت

تمرین 2 : پروتکل HTTP

1-2: تفاوت HTTP1.1 با HTTP2

HTTP2 نسخه جدید HTTP1.1 است که بر اساس پروتکل SPDY گوگل به منظور افزایش سرعت به سرویس صفحات وب طراحی شده است.

HTTP2 جایگزینی برای HTTP1.1 نیست. فقط یک پسوند است که کلیه مفاهیم مانند روش‌های HTTP، کدهای وضعیت، آدرس‌ها و فیلدهای Header یکسان هستند.



HTTP/1.1

دارای یکسری مشکلات است و این در حالی است که این پروتکل در زمانی ساخته شد که حجم صفحات وب به‌ ندرت از 100 کیلو بایت تجاوز می‌کرد، زبان CSS هنوز فراگیر نشده بود اما زبان JavaScript تازه پایش به وب باز شده بود ولی صرفاً برای یکسری اِفکت‌ ساده و یا وَلیدیشن فرم‌ استفاده می‌شد. زمانی که این پروتکل عرضه شد، پهنای باند بالا، فناوری ای‌جکس، وب اپلیکیشن‌های تک‌صفحه‌ای و فریمورک‌های سمت کاربر هنوز وجود خارجی پیدا نکرده بودند و در واقع تحت هیچ عنوان امکان ارسال بیش از چند ریکوئست به سرور برای اجرای کامل یک صفحهٔ وب در مرورگر کاربران وجود نداشت!

تفاوت‌های کلیدی HTTP2 و HTTP1.1

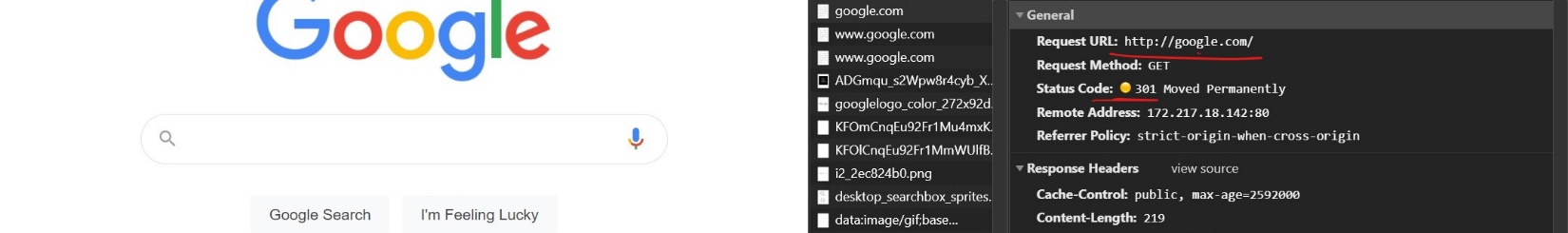
1. نسخهٔ HTTP2 حاوی داده‌های باینری (دودویی) است HTTP1.1 از داده‌های متنی استفاده می‌کند و این در حالی است که داده‌های متنی به طور کلی در سراسر شبکه از بازدهی کمتری نسبت به داده‌های باینری برخوردارند.
2. هدرهای HTTP2 فشرده‌ شده هستند: به طور کلی منظور از Header اطلاعاتی است که در پاسخ به یک ریکوئست ارسال می‌شود که شامل دیتا، مبدأ، نوع، حجم، مدت زمان کَش و موارد دیگر است. برخلاف HTTP1.1، این داده‌ها در نسخهٔ HTTP2 فشرده‌سازی می‌شوند تا پرفورمنس ارتقاء یابد.
3. نسخهٔ HTTP2 اصطلاحاً Asynchronous (نامتقارن) است در HTTP1.1، سرور باید به همان ترتیبی که ریکوئست‌ها را دریافت کرده است، ریسپانس‌ها را ارسال کند اما نسخهٔ HTTP2 اصطلاحاً Asynchronous است؛ بنابراین پاسخ‌های سریع‌تر و در عین حال با حجم کمتری می‌تواند در زمان کوتاه‌تری از سمت سرور ارسال شود.
4. نسخه HTTP2 مولتی‌پلکس است، در HTTP1.1، فقط یک درخواست روی یک کانکشن اینترنتی TCP در آن واحد می‌تواند به کار گرفته شود، HTTP2 اجازه ارسال چندین ریکوئست (درخواست) و دریافت ریسپانس (پاسخ) از سمت سرور را به طور هم‌ زمان(Parallel) بر روی یک کانکشن امکان‌پذیر می‌سازد

تمرین 2-2 :

کد های وضعیت سری 2 کدهایی هستند که نشان دهنده چراغ سبز و موفق بودن ریسپانس هایی است که از سمت سرور برای کاربر ارسال میشود اما کدهای سری 4 و 5 به معنی قرمز بودن و ایجاد خطا در پاسخ سرور به ما میباشد که میتوان به خطای 404 پیدا نشدن صفخه اشاره کرد

در این بین کدهایی داریم ک وضعیت ان ها با عدد 3 شروع میشوند این کدها نه وضعیت سبز ن وضعیت قرمز دارند بلکه وضعیتی زرد دارند به معنای راهنمایی یا هدایت مجدد از سمت سرور هستند ، به طور مثال وقتی شما ادرس <http://google.com> را جستجو کنید میبینید که یک کد وضعیت با شماره 301 برای شما ارسال میشود و در نهایت شما به ادرس <https://google.com> هدایت شده اید.

اتفاقی که اینجا میفتد یک redirect است یعنی سرور گوگل درخواست شما را گرفته ولی شما را به سرور جدید خودش که از نوع https و رمز گذاری شده است انتقال داده است



انواع وضعیت های 3xx

301 انتقال دائم (معروف به ریدایرکت 301) Moved Permanently

302 تغییر مسیر موقت Found

303 دیگری را ببینیدSee Other

304 اصلاح نشده Not Modified

تمرین 2-3 :

Host :

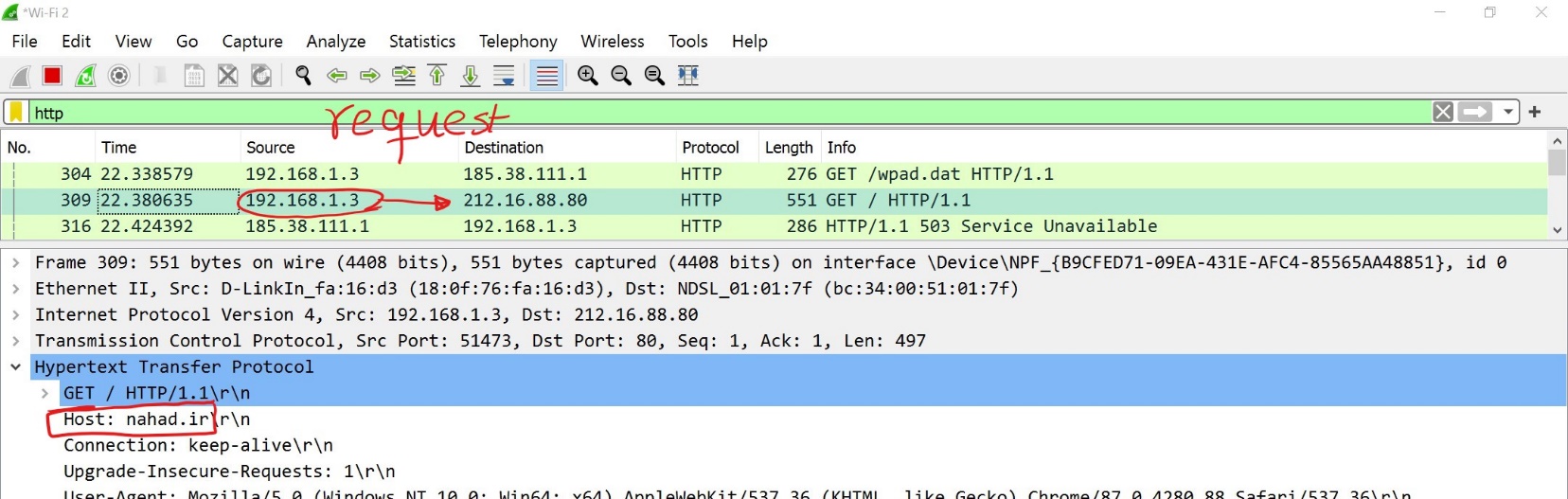
The Host request header specifies the host and port number of the server to which the request is being sent.

If no port is included, the default port for the service requested (e.g., 443 for an HTTPS URL, and 80 for an HTTP URL) is implied.

A Host header field must be sent in all HTTP/1.1 request messages. A 400 (Bad Request) status code may be sent to any HTTP/1.1 request message that lacks a Host header field or that contains more than one.

Syntax

Host: <host>:<port>



Referer :

The Referer request header contains the address of the page making the request.

The Referer header allows servers to identify where people are visiting them from and may use that data for analytics, logging, or optimized caching, for example.

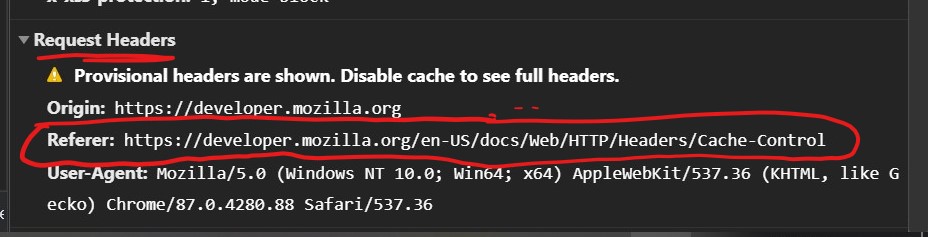
A Referer header is not sent by browsers if:

The referring resource is a local "file" or "data" URI.

An unsecured HTTP request is used and the referring page was received with a secure protocol (HTTPS).

Syntax

Referer: <url>



Accept-Encoding :

The Accept-Encoding request HTTP header advertises which content encoding, usually a compression algorithm, the client is able to understand. Using content negotiation, the server selects one of the proposals, uses it and informs the client of its choice with the Content-Encoding response header.

Syntax

Accept-Encoding: gzip

Accept-Encoding: compress

Accept-Encoding: deflate

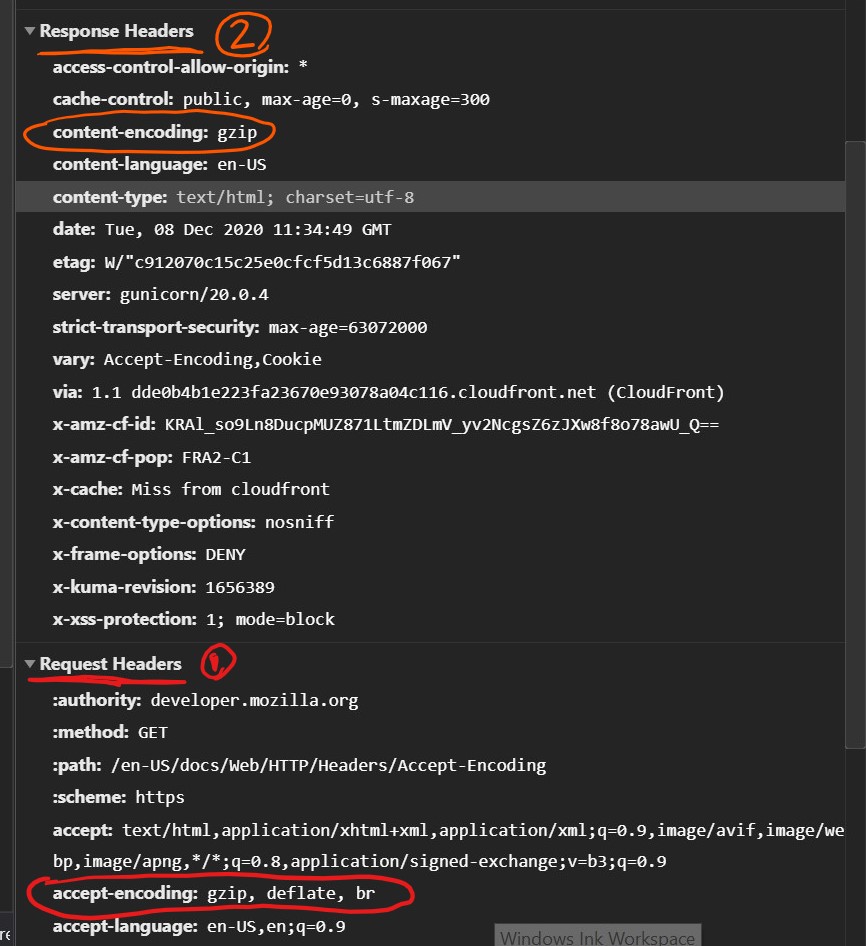
Accept-Encoding: br

Accept-Encoding: identity

Accept-Encoding: \*

// Multiple algorithms, weighted with the [quality value](https://developer.mozilla.org/en-US/docs/Glossary/Quality_Values) syntax:

Accept-Encoding: deflate, gzip; q=1.0, \*;q=0.5



Content-type:

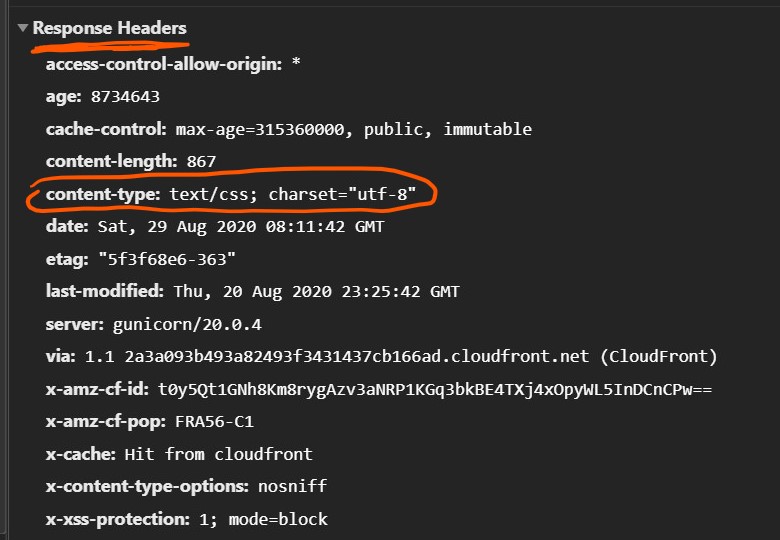
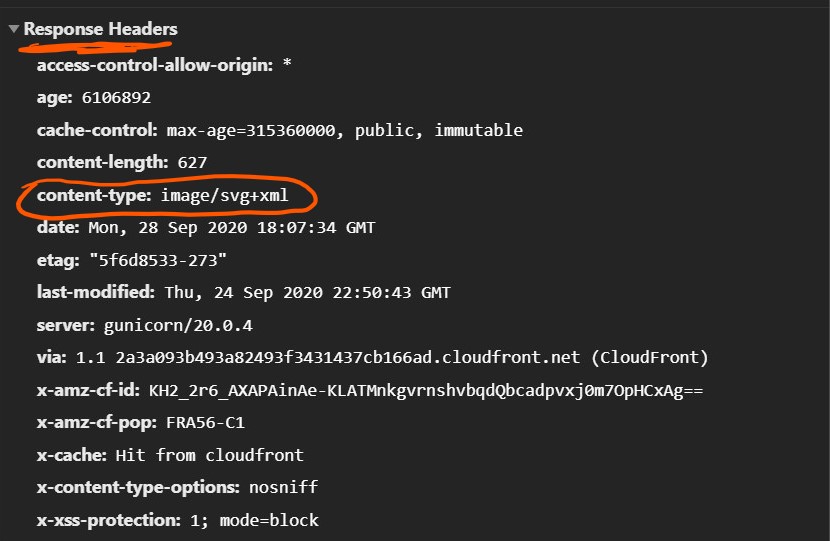
The Content-Type entity header is used to indicate the media type of the resource.

In responses, a Content-Type header tells the client what the content type of the returned content actually is.

Syntax

Content-Type: text/html; charset=UTF-8

Content-Type: multipart/form-data; boundary=something

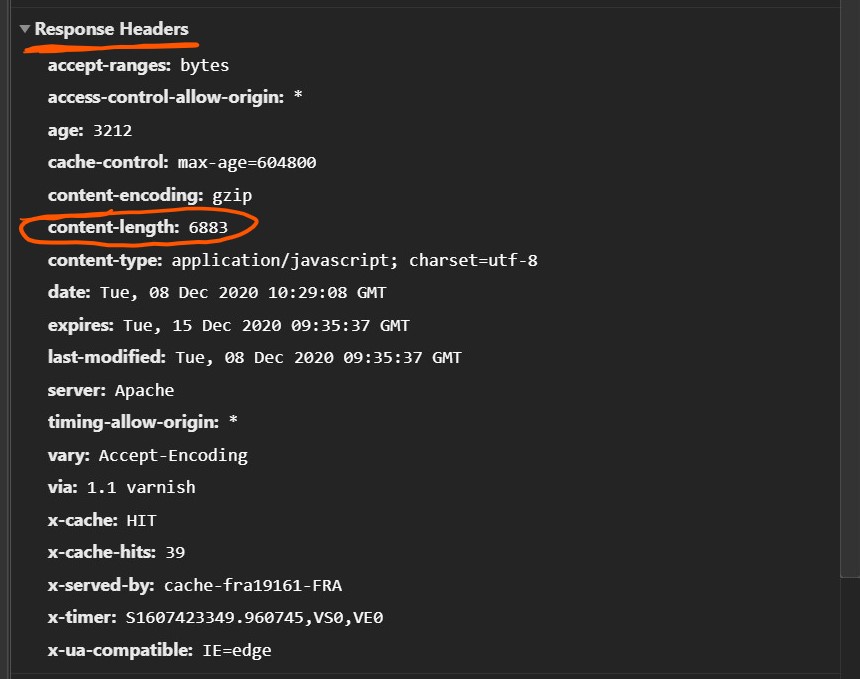


Content-Length:

The Content-Length entity header indicates the size of the entity-body, in bytes, sent to the recipient.

Syntax

Content-Length: <length>



Content-Range:

The Content-Range response HTTP header indicates where in a full body message a partial message belongs.

Syntax

Content-Range: <unit> <range-start>-<range-end>/<size>

Content-Range: <unit> <range-start>-<range-end>/\*

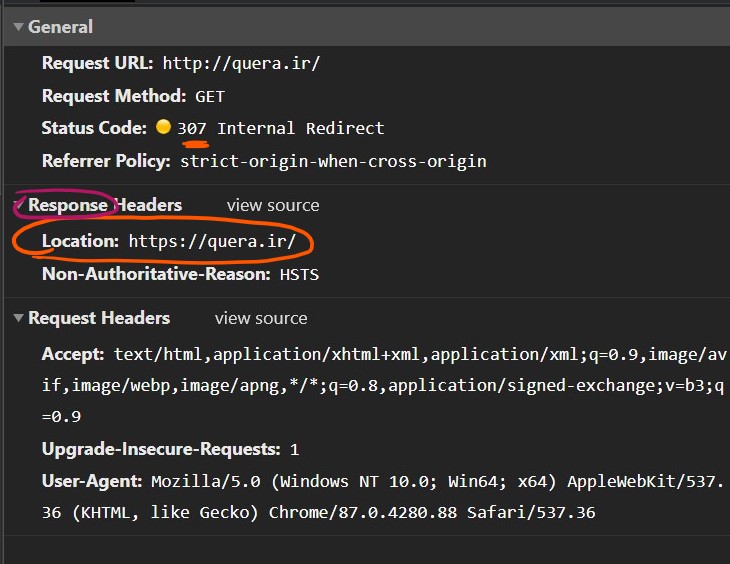
Content-Range: <unit> \*/<size>

Examples

Content-Range: bytes 200-1000/67589

Location:

The Location response header indicates the URL to redirect a page to. It only provides a meaning when served with a 3xx (redirection) or 201 (created) status response.

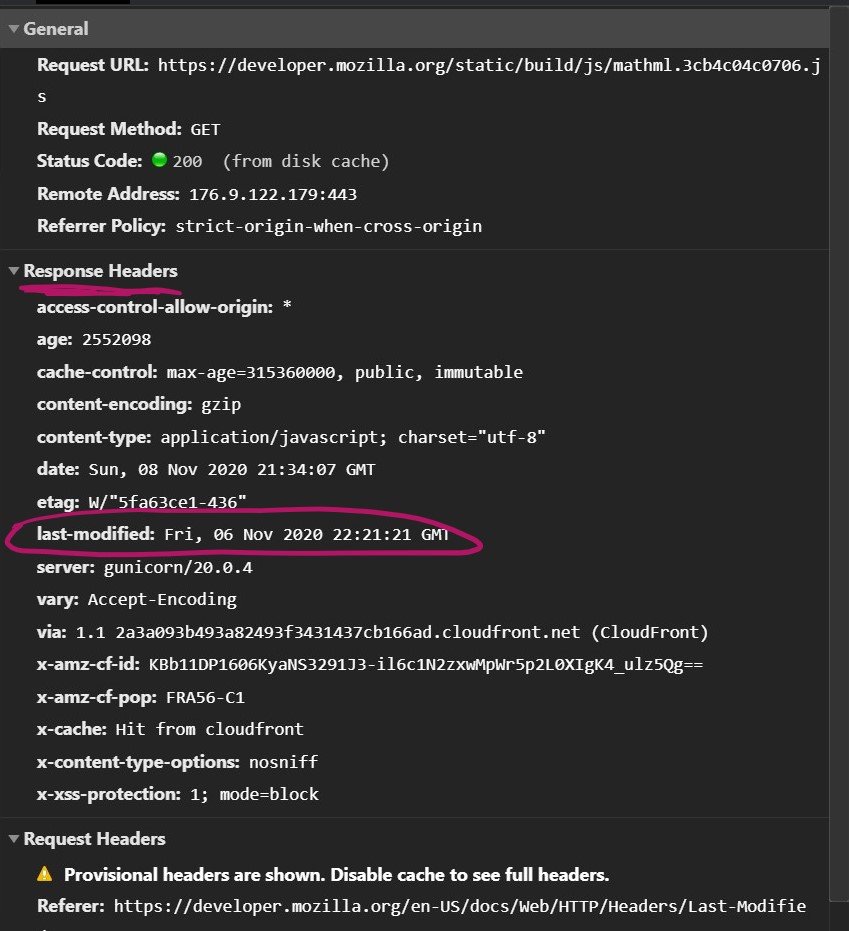


Last-Modified:

The Last-Modified response HTTP header contains the date and time at which the origin server believes the resource was last modified. It is used as a validator to determine if a resource received or stored is the same.

Syntax

Last-Modified: <day-name>, <day> <month> <year> <hour>:<minute>:<second> GMT



Cache-Control:

The Cache-Control HTTP header holds directives (instructions) for caching in both requests and responses. A given directive in a request does not mean the same directive should be in the response.

Syntax

Caching directives have the following rules to be valid:

* Case-insensitive, but lowercase is recommended.
* Multiple directives are comma-separated.
* Some directives have an optional argument, which can be either a token or a quoted-string. (See spec for definitions)

### Cache request directives

Standard Cache-Control directives that can be used by the client in an HTTP request.

Cache-Control: max-age=<seconds>

Cache-Control: max-stale[=<seconds>]

Cache-Control: min-fresh=<seconds>

Cache-Control: no-cache

Cache-Control: no-store

Cache-Control: no-transform

Cache-Control: only-if-cached

### Cache response directives

Standard Cache-Control directives that can be used by the server in an HTTP response.

Cache-Control: must-revalidate

Cache-Control: no-cache

Cache-Control: no-store

Cache-Control: no-transform

Cache-Control: public

Cache-Control: private

Cache-Control: proxy-revalidate

Cache-Control: max-age=<seconds>

Cache-Control: s-maxage=<seconds>

Examples

Cache-Control: no-store

