# COMS W4156 Advanced Software Engineering (ASE)

October 14, 2021

shared google doc for discussion during class

# More Writing Code the Right Way: Secure Coding

Style checkers look for “patterns” in the code and flag code that does not conform to required patterns (*always do*)

Most also look for code smells - anti-patterns to be avoided (*never do*)

Some patterns and anti-patterns are specifically concerned with security

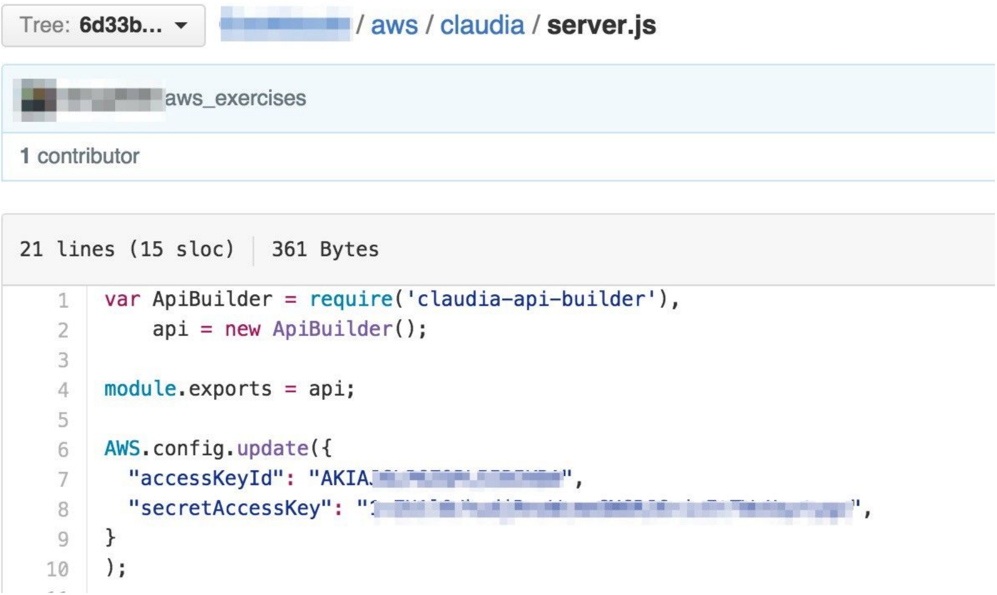
Example secure coding guidelines

# What is Insecure Coding?

Simplest example: secret keys, tokens or passwords embedded in code

String constants containing domain-specific patterns - in this case access key patterns known to appear in AWS configurations

[CWE-798: Use of Hard-coded Credentials]()



[redacted] / aws / claudia / **server.js**

[redacted] aws_exercises

1 contributor

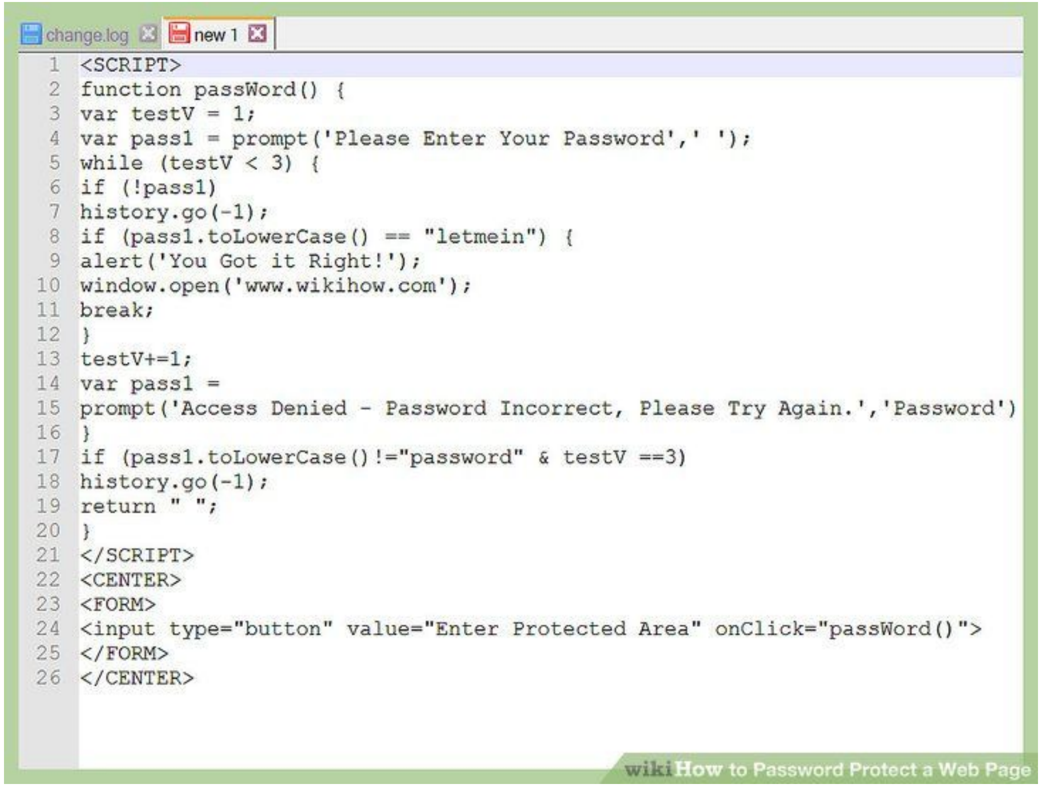
21 lines (15 sloc) | 361 Bytes

```
var ApiBuilder = require('claudia-api-builder'),
    api = new ApiBuilder();

module.exports = api;

AWS.config.update({
  "accessKeyId": "AKIA[redacted]",
  "secretAccessKey": "[redacted]",
}
);
```

# What is Wrong?



```
<SCRIPT>
function passWord() {
var testV = 1;
var pass1 = prompt('Please Enter Your Password',' ');
while (testV < 3) {
if (!pass1)
history.go(-1);
if (pass1.toLowerCase() == "letmein") {
alert('You Got it Right!');
window.open('www.wikihow.com');
break;
}
testV+=1;
var pass1 =
prompt('Access Denied - Password Incorrect, Please Try Again.','Password')
}
if (pass1.toLowerCase()!="password" & testV ==3)
history.go(-1);
return " ";
}
</SCRIPT>
<CENTER>
<FORM>
<input type="button" value="Enter Protected Area" onClick="passWord()">
</FORM>
</CENTER>
```

wikiHow to Password Protect a Web Page

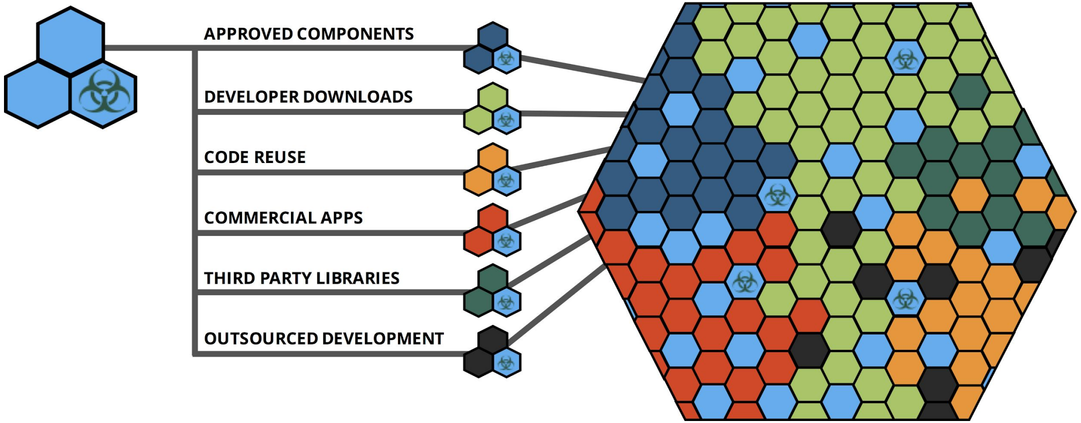
Found in a tutorial on how to add password protection to a page using Javascript

CWE-798: Use of Hard-coded Credentials

# Using Third-Party Libraries with Known Vulnerabilities

Detected by comparing code resources to databases of known security vulnerabilities

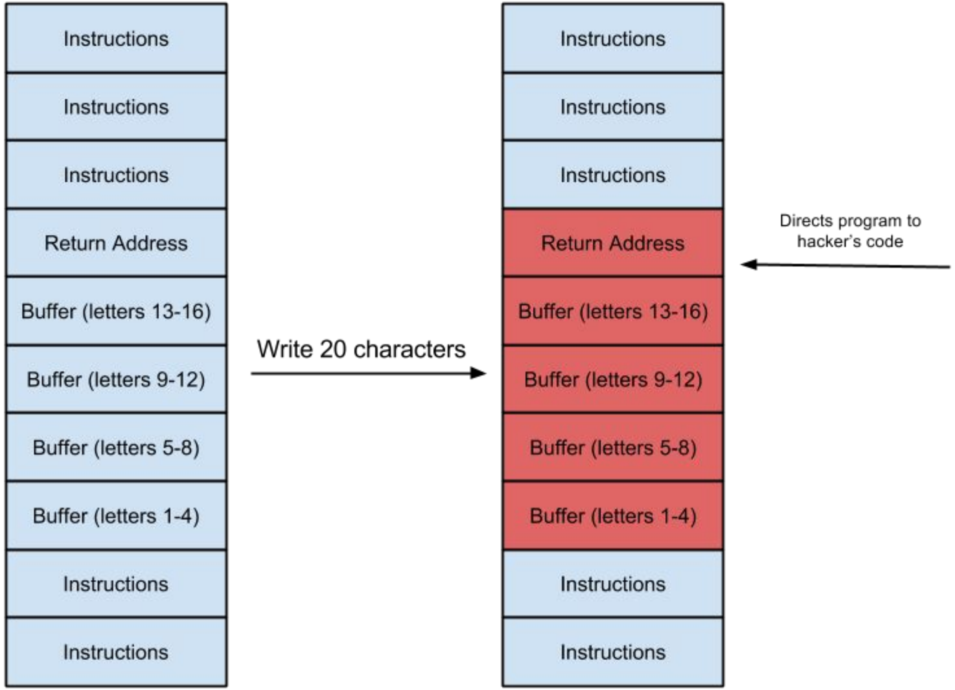
[A9:2017-Using Components with Known Vulnerabilities]



# Everyone’s Favorite Vulnerability: Buffer Overflows

A bug finder for unmanaged languages (notably C/C++) should look for code patterns that check array bounds, to prevent buffer overflow

https://cwe.mitre.org/ -> “buffer overflow”



Instructions
Instructions
Instructions
Return Address
Buffer (letters 13-16)
Buffer (letters 9-12)
Buffer (letters 5-8)
Buffer (letters 1-4)
Instructions
Instructions

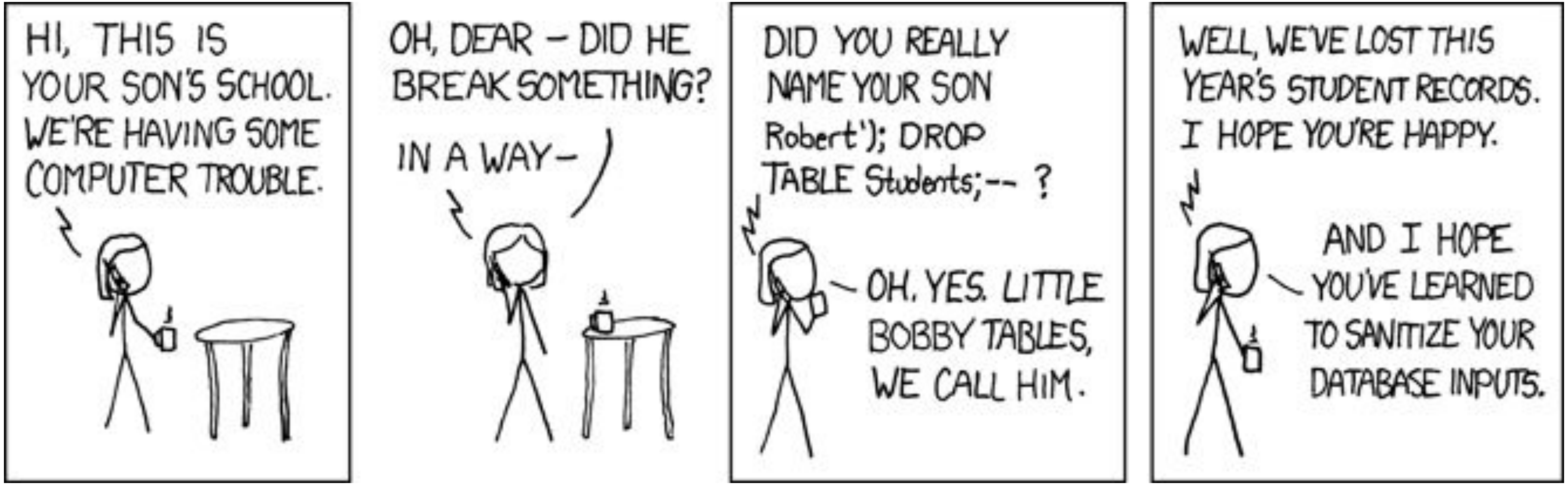
Write 20 characters

Instructions
Instructions
Instructions
Return Address
Buffer (letters 13-16)
Buffer (letters 9-12)
Buffer (letters 5-8)
Buffer (letters 1-4)
Instructions
Instructions

Directs program to hacker’s code

# Unsanitized user inputs

Wen user inputs are passed to database, API, logger, external system, etc.


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

# Explanation

The school apparently stores student names in a database table called Students. When a new student enrolls, the school inserts his/her name into this table. The code doing the insertion might look like:

```
    $sql = "INSERT INTO Students (Name)
            VALUES ('" . $studentName . "');";
    execute_sql($sql);
```

This code first creates a string containing an SQL INSERT statement. The content of the $studentName variable is glued into the SQL statement. Then the code sends the resulting SQL statement to the database. Untrusted user input, i.e., the content of $studentName, becomes part of the SQL statement.

# Explanation continued

Say the user input is “Sarah”, then the SQL is:

```
INSERT INTO Students (Name) VALUES ('Sarah')’
```

This inserts Sarah into the Students table.

Now say the user input is “Robert'); DROP TABLE Students;--” then the SQL statement becomes

```
INSERT INTO Students (Name) VALUES ('Robert');
DROP TABLE Students;--');
```

This first inserts Robert into the Students table. But since the INSERT statement is followed by a DROP TABLE statement , next the entire Students table is deleted.

# Sanitizing User Inputs

However, what it means to “sanitize” user inputs is not well-defined

The only guaranteed way to avoid an “SQL injection” attack is to use “prepared statements”

Every programming language commonly used with databases has some facility for passing user inputs as parameters to prepared SQL statements

# Check for all errors/exceptions

“*Software flaws are security flaws*” (Rebecca Wright)

The conventional way for hackers to exploit a software flaw (bug) is to craft invalid input that makes it past input validity checks, so it’s crucial to handle *every* error code or exception from each API call

Long list of Linux error codes: errno - number of last error

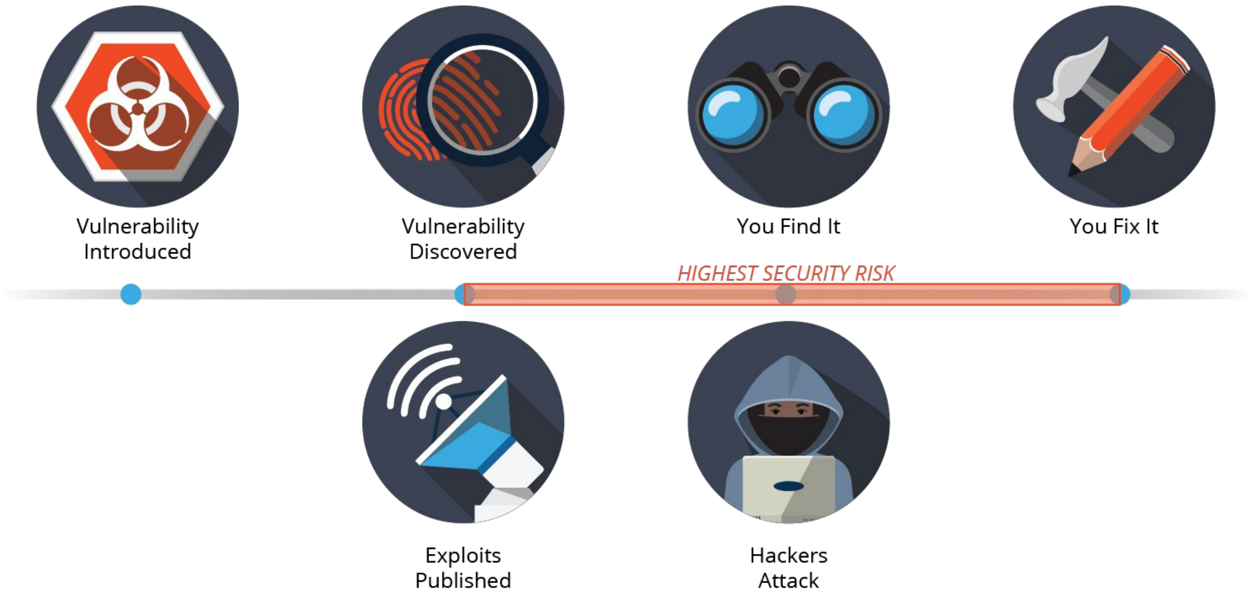
https://cwe.mitre.org/ -> “handling status codes”

# Errors vs Exceptions

- A user entering the wrong data is not exceptional and does not need to be handled with an exception. Simple checks, on both front-end and back-end, can address true user errors - and show meaningful error messages. But never rely on the front-end to protect the back-end
- A file won’t open and is throwing FileLoadException or FileNotFoundException. This is an exceptional situation that should be handled by catching the exception with appropriate processing code - and, again, show meaningful error messages to the user

➢ If the “user” is - or could be - other code, then the server or service must return meaningful status codes according to an agreed-upon protocol, and the user/caller code should branch on status codes

# Vulnerability Lifecycle



Vulnerability Introduced

Vulnerability Discovered

You Find It

You Fix It

HIGHEST SECURITY RISK

Exploits Published

Hackers Attack

# What are the Goals of Secure Coding?

Confidentiality: Data not leaked

Integrity: Data not modified

Availability: Data is accessible when needed

Authenticity: Data origin cannot be spoofed

# In-Class Exercise

Recall the CONTAIN app we have been working on, **CON**tact **T**racing for instruction **A**ss**I**sta**N**ts, which does “contact tracing” based on IA (instruction assistant) connections. An IA for xxx course is a contact of all the other IAs for xxx as well as for every student who takes xxx. Since an IA is also a student, they are a contact for all other students in every course they take as well as for all the IAs of those courses.

We would like to maintain security: Confidentiality (Data not leaked), Integrity (Data not modified), Availability (Data is accessible when needed), Authenticity (Data origin cannot be spoofed)

Pick one of these four types, figure out how it is relevant to CONTAIN and its interactions with SSOL, CANVAS, and drone-based attendance-recording service APIs, and devise a plan for coding and testing CONTAIN intended to prevent security breaches of that type timer...

# Common Weakness Enumeration

Bad Coding Practices

API / Function Errors

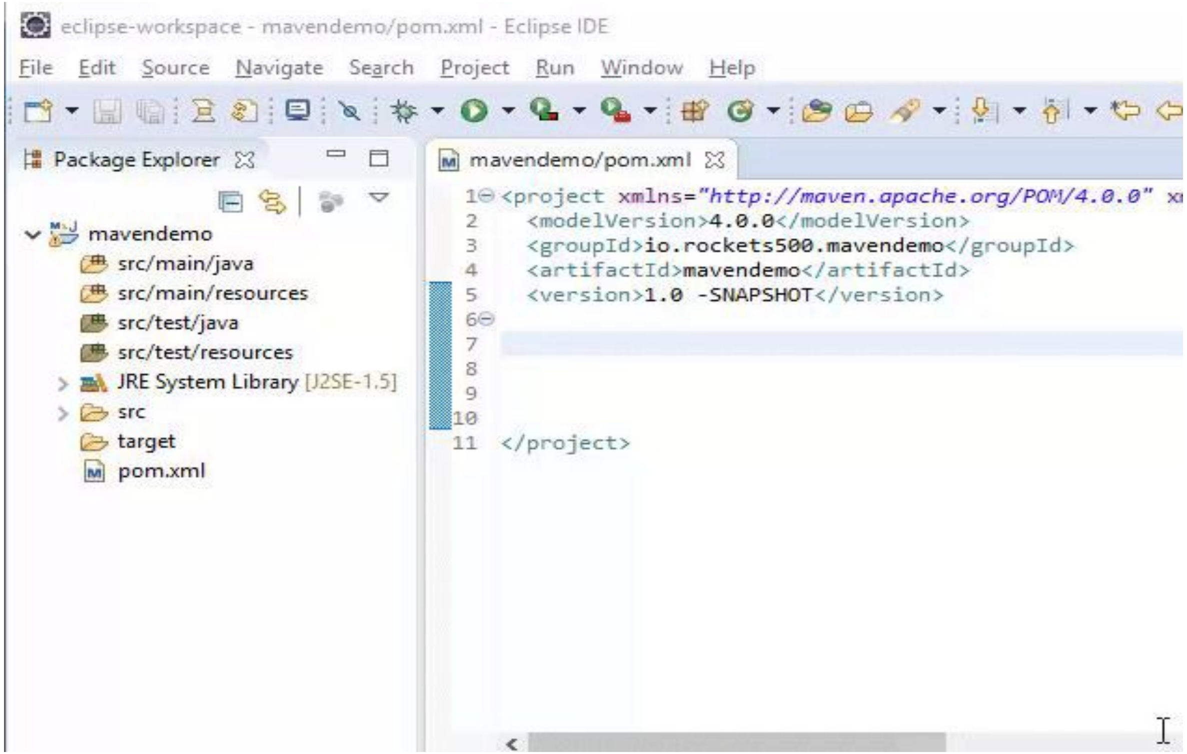
# Integrated Development Environment (IDE)

Comprehensive support for coding, testing and debugging

Intended for projects with *many* files, make it easy to move between interdependent files

Many IDEs support code completion based on context of entire project

Typical features include refactoring, integration with version control, code search, automated testing, test coverage tracking, interactive debugging, style checking, various static analysis, software metrics…



```
<project xmlns="http://maven.apache.org/POM/4.0.0" x
  <modelVersion>4.0.0</modelVersion>
  <groupId>io.rockets500.mavendemo</groupId>
  <artifactId>mavendemo</artifactId>
  <version>1.0 -SNAPSHOT</version>





</project>
```

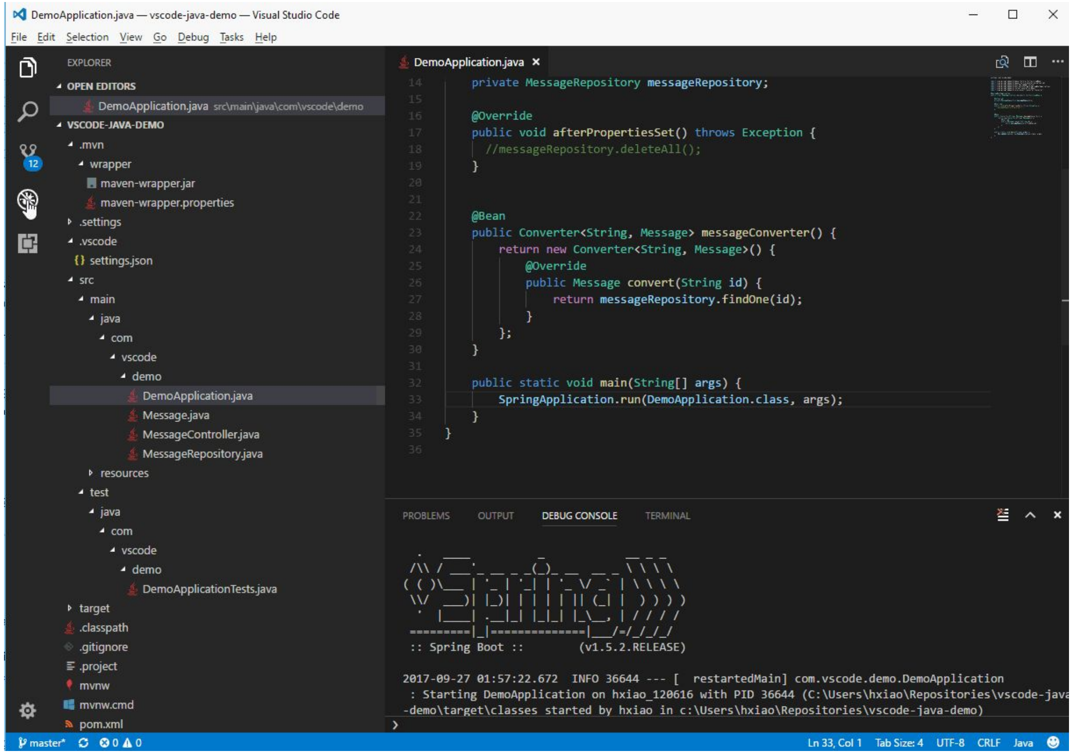
# Code Editor

Special text editor that “knows” you are writing code in a particular programming language

Code editors typically highlight language keywords, catch syntax errors, and compile and run programs

Code editors are usually fast and easy to use, whereas IDEs can be slow to startup (since they load the entire project) and their UIs can be daunting

No clear dividing line between what is IDE vs. code editor



```java
    private MessageRepository messageRepository;

    @Override
    public void afterPropertiesSet() throws Exception {
      //messageRepository.deleteAll();
    }


    @Bean
    public Converter<String, Message> messageConverter() {
        return new Converter<String, Message>() {
            @Override
            public Message convert(String id) {
                return messageRepository.findOne(id);
            }
        };
    }

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.2.RELEASE)

2017-09-27 01:57:22.672  INFO 36644 --- [  restartedMain] com.vscode.demo.DemoApplication
 : Starting DemoApplication on hxiao_120616 with PID 36644 (C:\Users\hxiao\Repositories\vscode-java
-demo\target\classes started by hxiao in c:\Users\hxiao\Repositories\vscode-java-demo)
>
```

# What you should do

Strongly recommended (not required) for team members to use the same IDE or code editor

Install same plugins, share vocabulary and learn from each other

But if you insist on using NotePad, TextEdit or vim, good luck!

# Team Project

You will soon hear from your (tentative) IA mentor

Assignment T2: Revised Project Proposal due October 25

Deadline is awhile off to allow time for IA mentor assignments and then for you to meet with your IA mentor

You can and should start working on this now, but you cannot finalize until after you meet with your IA mentor

You can and should start coding as soon as possible after the meeting