… an (un)expected journey

The fellowship

Balázs Vojtek

Gabriel Szabó

The fellowship

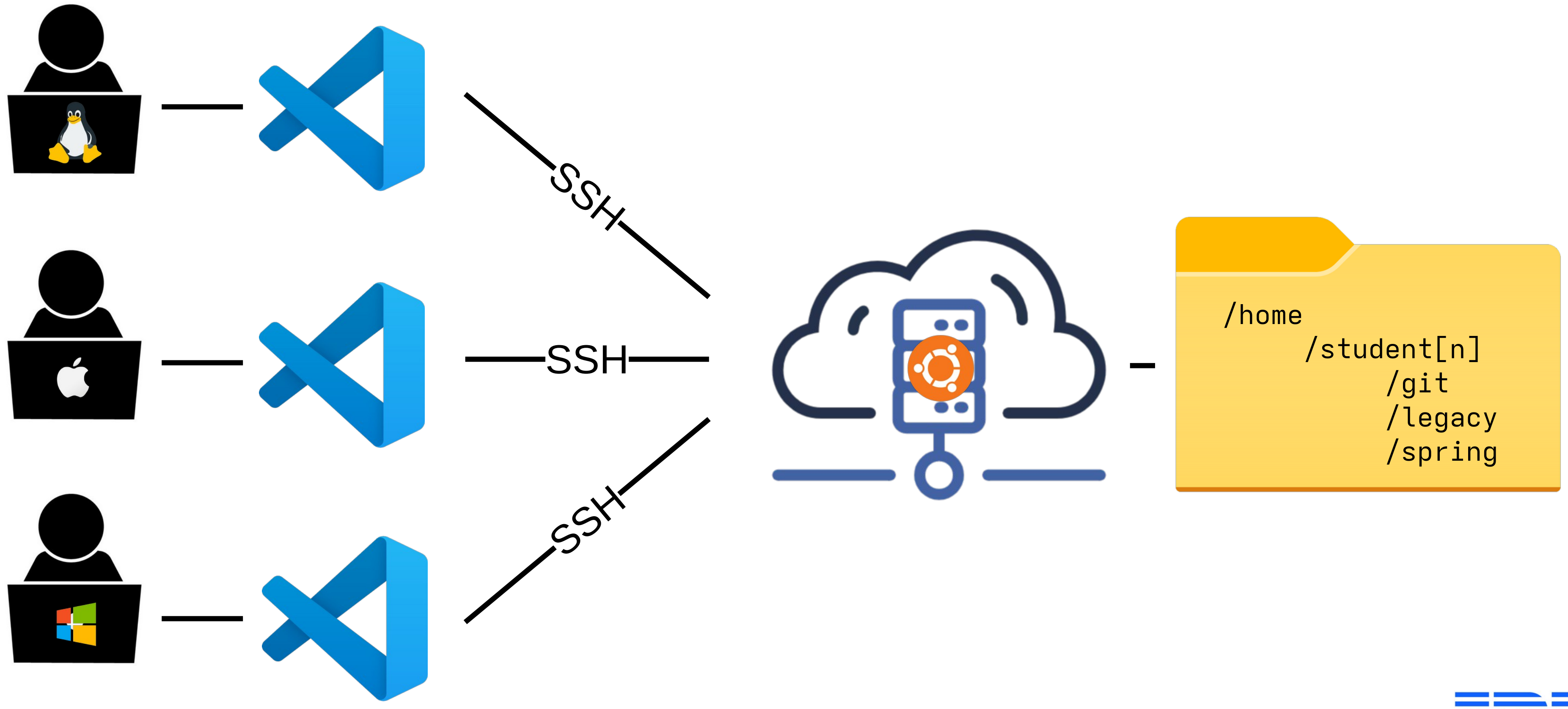
Viktor Jandák

Ivan Bohuš

Agenda

- Setup
- Git
- Java recap
- Sockets and networking
- JDBC and H2
- Configuration
- Wrap up

Setup



Setup

- Use your ID to connect to the remote server
- Use the given port range during development
- Test access via SSH

- Home folder:
  `/home/student[n]`
- Remote folders
  - Git intro: `~/git`
  - Legacy: `~/legacy`
  - Spring: `~/spring`

Student	ID	Ports	Password
Aleksandr Rakov	student01	8100 - 8199	nala5-ku
Alex Haščík	student02	8200 - 8299	trev3-mo
Vladyslav Pehushyn	student03	8300 - 8399	pako7-li
Mykhailo Pavlov	student04	8400 - 8499	sedu4-ra

Setup

- Download Visual Studio Code

  https://code.visualstudio.com/download

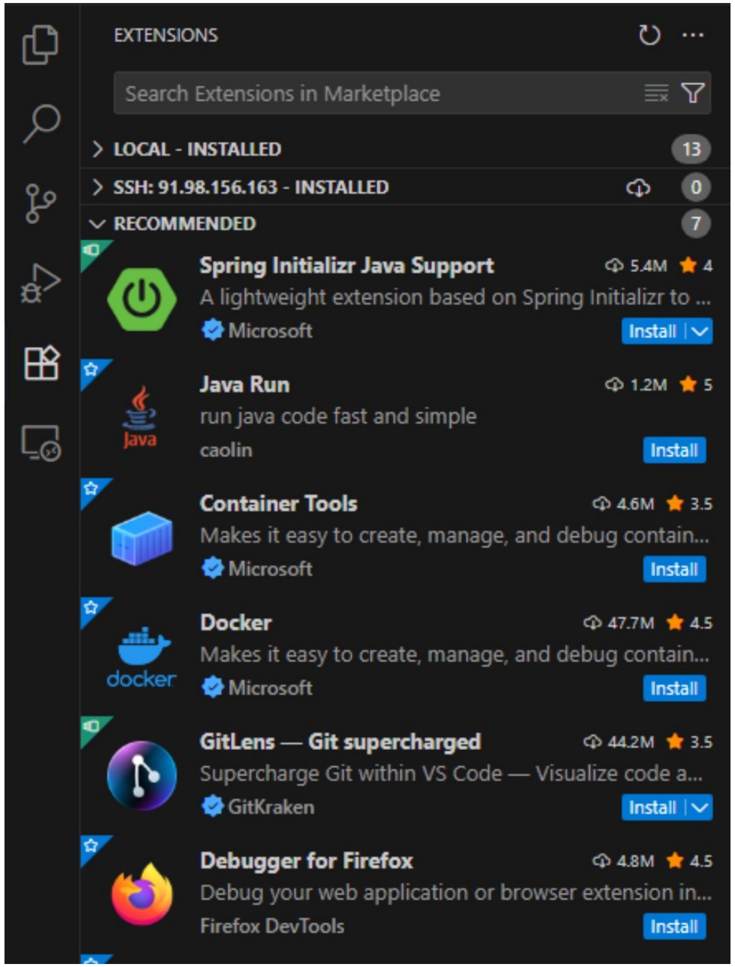
- Add plugin “Remote – SSH” (ms-vscode-remote.remote-ssh)

- Connect to the remote server

  - `ssh student[n]@91.98.156.163`

Setup

- Install recommended plugins
  - checkout workspace
  - `.vscode/extension.json`
  - select recommended plugins


EXTENSIONS
Search Extensions in Marketplace
LOCAL - INSTALLED 13
SSH: 91.98.156.163 - INSTALLED 0
RECOMMENDED 7

Spring Initializr Java Support — 5.4M ★ 4
A lightweight extension based on Spring Initializr to ...
Microsoft — Install

Java Run — 1.2M ★ 5
run java code fast and simple
caolin — Install

Container Tools — 4.6M ★ 3.5
Makes it easy to create, manage, and debug contain...
Microsoft — Install

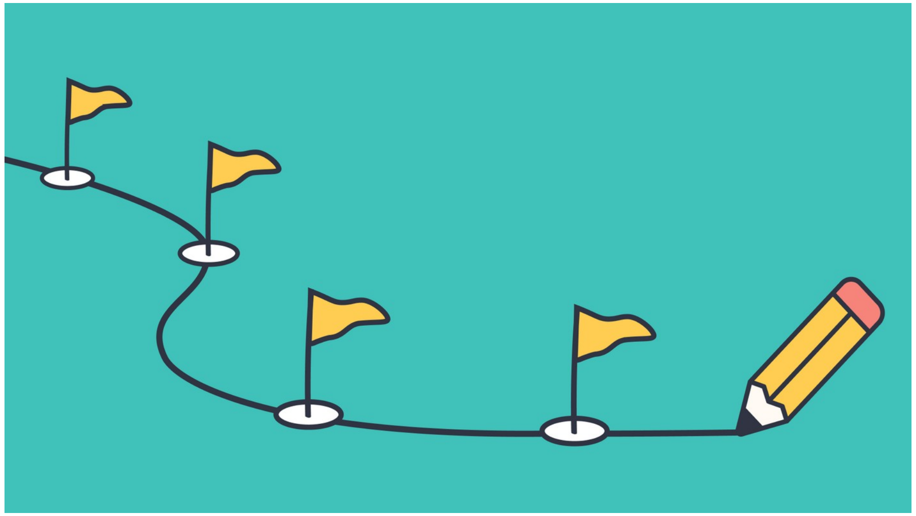
Docker — 47.7M ★ 4.5
Makes it easy to create, manage, and debug contain...
Microsoft — Install

GitLens — Git supercharged — 44.2M ★ 3.5
Supercharge Git within VS Code — Visualize code a...
GitKraken — Install

Debugger for Firefox — 4.8M ★ 4.5
Debug your web application or browser extension in...
Firefox DevTools — Install

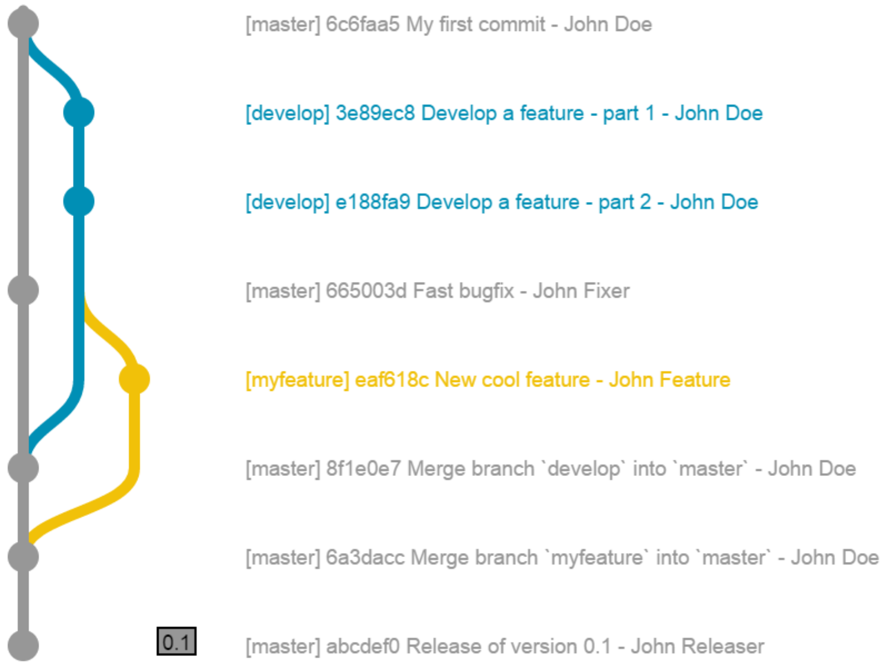
Learning goals

- Recap basic knowledge
  - Git
  - Java
  - Instance flow
- Develop a simple app
  - Plain Java
  - Maven
- Test the given app
  - Manual
  - Scripted



Git in a nutshell

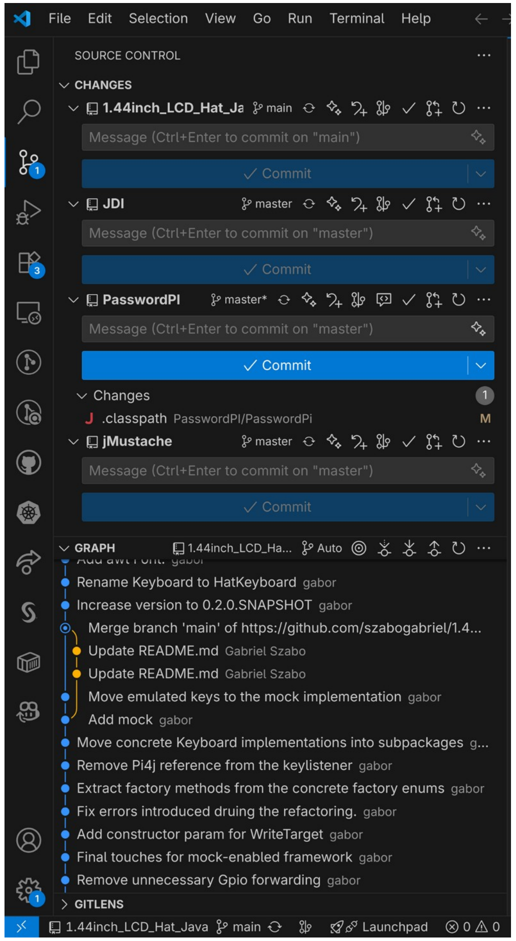
- De-facto standard
- Integration with IDE and CLI
- Different flavors available free of charge
- Different strategies
- Navigate through commits (hashes)
- Pitfalls
  - Keystores, secrets and passwords
  - Long living feature branches



[master] 6c6faa5 My first commit - John Doe
[develop] 3e89ec8 Develop a feature - part 1 - John Doe
[develop] e188fa9 Develop a feature - part 2 - John Doe
[master] 665003d Fast bugfix - John Fixer
[myfeature] eaf618c New cool feature - John Feature
[master] 8f1e0e7 Merge branch `develop` into `master` - John Doe
[master] 6a3dacc Merge branch `myfeature` into `master` - John Doe
0.1
[master] abcdef0 Release of version 0.1 - John Releaser

Git + VS Code

- Plugin by default present in VS Code
- Provides a UI for working with GIT
- Top view – local changes
- Bottom view – remote changes



File Edit Selection View Go Run Terminal Help

SOURCE CONTROL

CHANGES

1.44inch_LCD_Hat_Ja main

Message (Ctrl+Enter to commit on "main")

✓ Commit

JDI master

Message (Ctrl+Enter to commit on "master")

✓ Commit

PasswordPI master*

Message (Ctrl+Enter to commit on "master")

✓ Commit

Changes 1

.classpath PasswordPI/PasswordPi M

jMustache master

Message (Ctrl+Enter to commit on "master")

✓ Commit

GRAPH 1.44inch_LCD_Ha... Auto

Add awt Font. gabor
Rename Keyboard to HatKeyboard gabor
Increase version to 0.2.0.SNAPSHOT gabor
Merge branch 'main' of https://github.com/szabogabriel/1.4...
Update README.md Gabriel Szabo
Update README.md Gabriel Szabo
Move emulated keys to the mock implementation gabor
Add mock gabor
Move concrete Keyboard implementations into subpackages g...
Remove Pi4j reference from the keylistener gabor
Extract factory methods from the concrete factory enums gabor
Fix errors introduced druing the refactoring. gabor
Add constructor param for WriteTarget gabor
Final touches for mock-enabled framework gabor
Remove unnecessary Gpio forwarding gabor

GITLENS

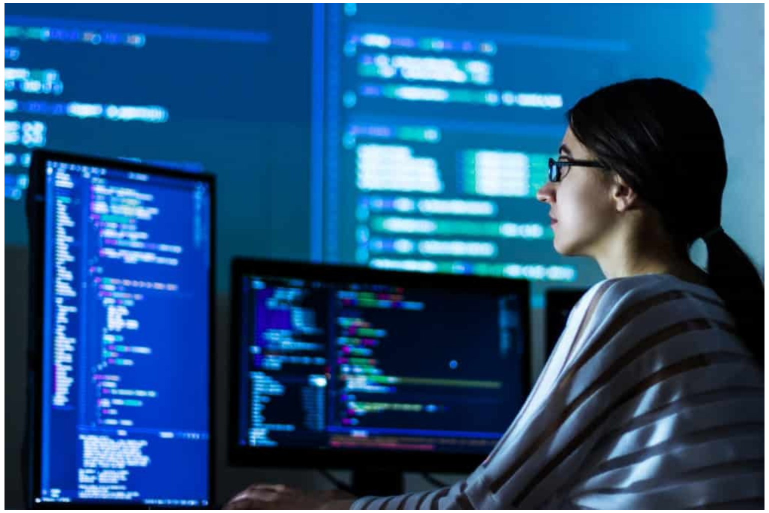
1.44inch_LCD_Hat_Java main Launchpad ⊗ 0 ⚠ 0

Core Git Workflow

- Create local copy
  - git clone
  - E.g.: `git clone https://github.com/szabogabriel/FotS_2025_git.git .`
  - Use '.' at the end to checkout to current folder
- Create new local branch
  - `git checkout -b feature/[name]`
- State of local branch
  - `git status`
- Add changes / files
  - `git add`
- Commit
  - `git commit -m "Message"`
- Push changes to the server
  - `git push`
- Receive changes
  - `git pull`
  - `git fetch origin`

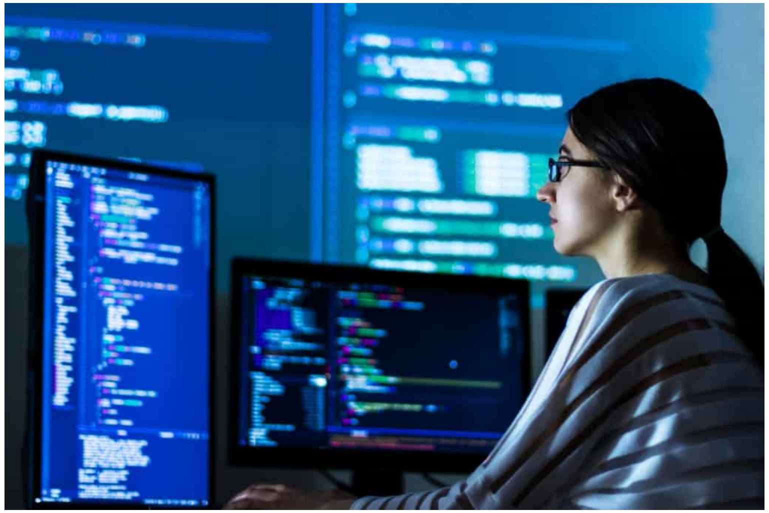
DEV: basic git workflow

1) Go to `https://github.com/settings/keys`
2) Configure git (username / keys)
   - Generate key:
     - CLI: `ssh-keygen -t ed25519 -C "your_mail@example.com"`
     - Windows: `putty`
   - Add public key to Github



DEV: basic git workflow

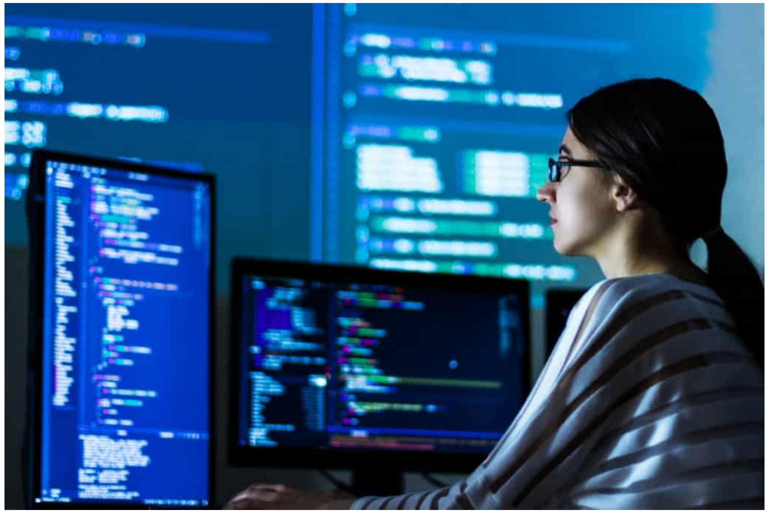
3) Clone the project repository
    - `https://github.com/szabogabriel/FotS_2025_git`
4) Create a local feature branch
5) Create a new file `student[n].md` (e.g. `student01.md`)
6) Commit and push changes
7) Create pull request



DEV: conflict & merge

8) Resolve conflicts

9) Commit and push again



Git quick reference

- Cheat sheet in the handout
- Tips and tricks
  - Don’t push secrets
  - Write good comments
  - Commit often – merge often
  - The first one to merge, doesn’t need to resolve conflicts

Java language and platform

- Strongly typed, object oriented language
- Compiled to bytecode, interpreted on Java Virtual Machine (JVM)
- Classes loaded dynamically into classpath
- Automatic memory management
  - Garbage collection
- Java memory model
  - Heap space
  - Perm space

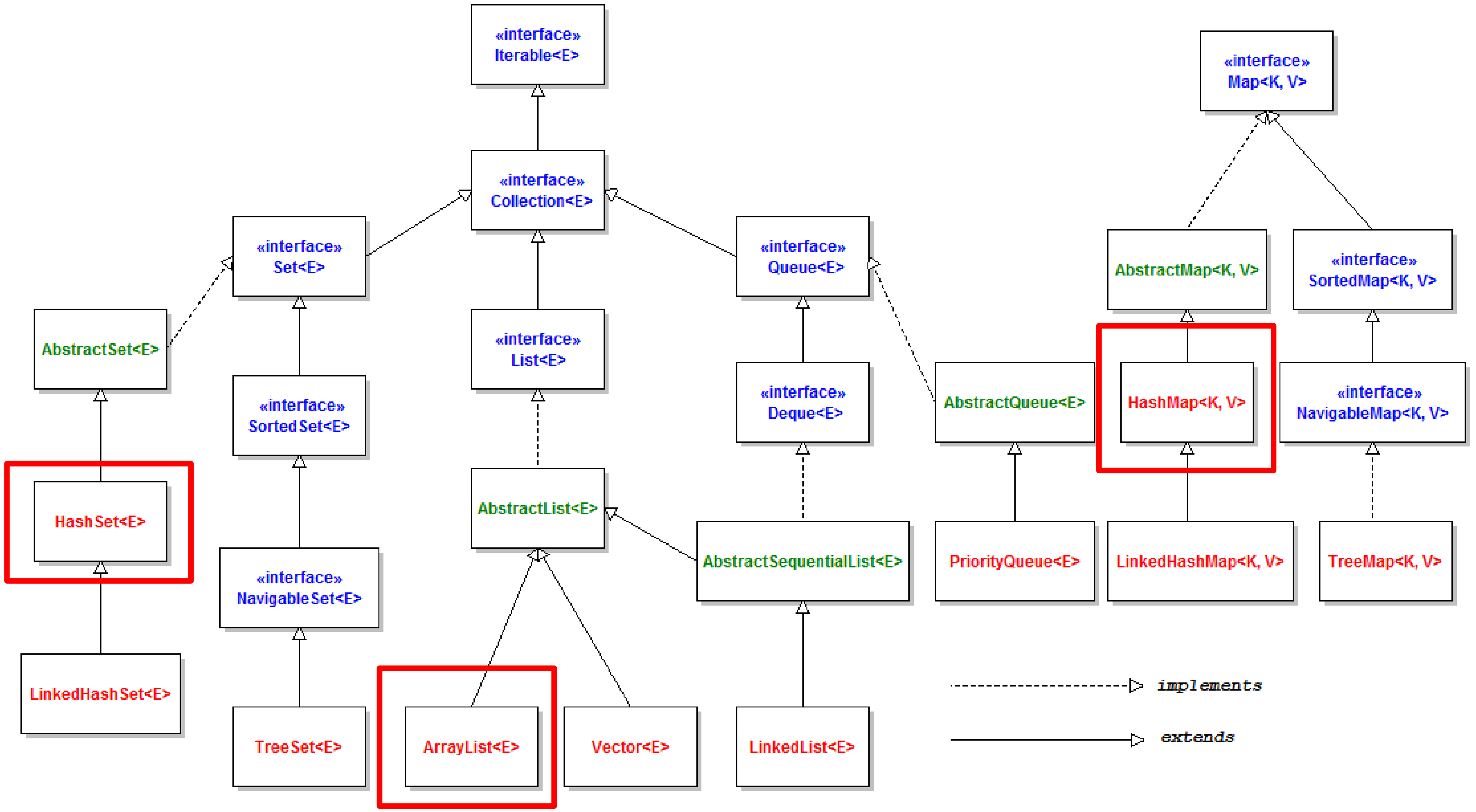
Java building blocks

```
class A {
    private static int i;
    int j;
    public static String
    getA() { return “A”; }
    protected String getB()
    { return “B”; }
}
```

Interfaces, abstract classes & enums

```
interface Runnable {
    void run();
}
abstract class X {
    protected void run();
    public x() {}
}
enum Status {
    NEW, IN_PROGRESS, DONE;
}
```

Collections to be used



Lambdas

- Lambdas:
  - Method without a name
  - `list.forEach(`**`e → handle(e)`**`);`
- Method reference

  `list.forEach(`**`System.out::println`**`);`
- Functional interfaces
  - `Predicate → boolean test(T val);`
  - `Consumer → void accept(T val);`
  - `Supplier → T get();`
  - `Function → R apply(T val);`

Threads

- Thread - class
  - Bound to system threads
    - Except for Virtual Threads
  - Basis for parallel execution
- Runnable
  - Functional interface
  - Parameter for Thread’s constructor
- ExecutorService
  - Interface for execution strategies
  - ThreadPoolExecutor, ForkJoinPool etc.
  - Won’t use it for now

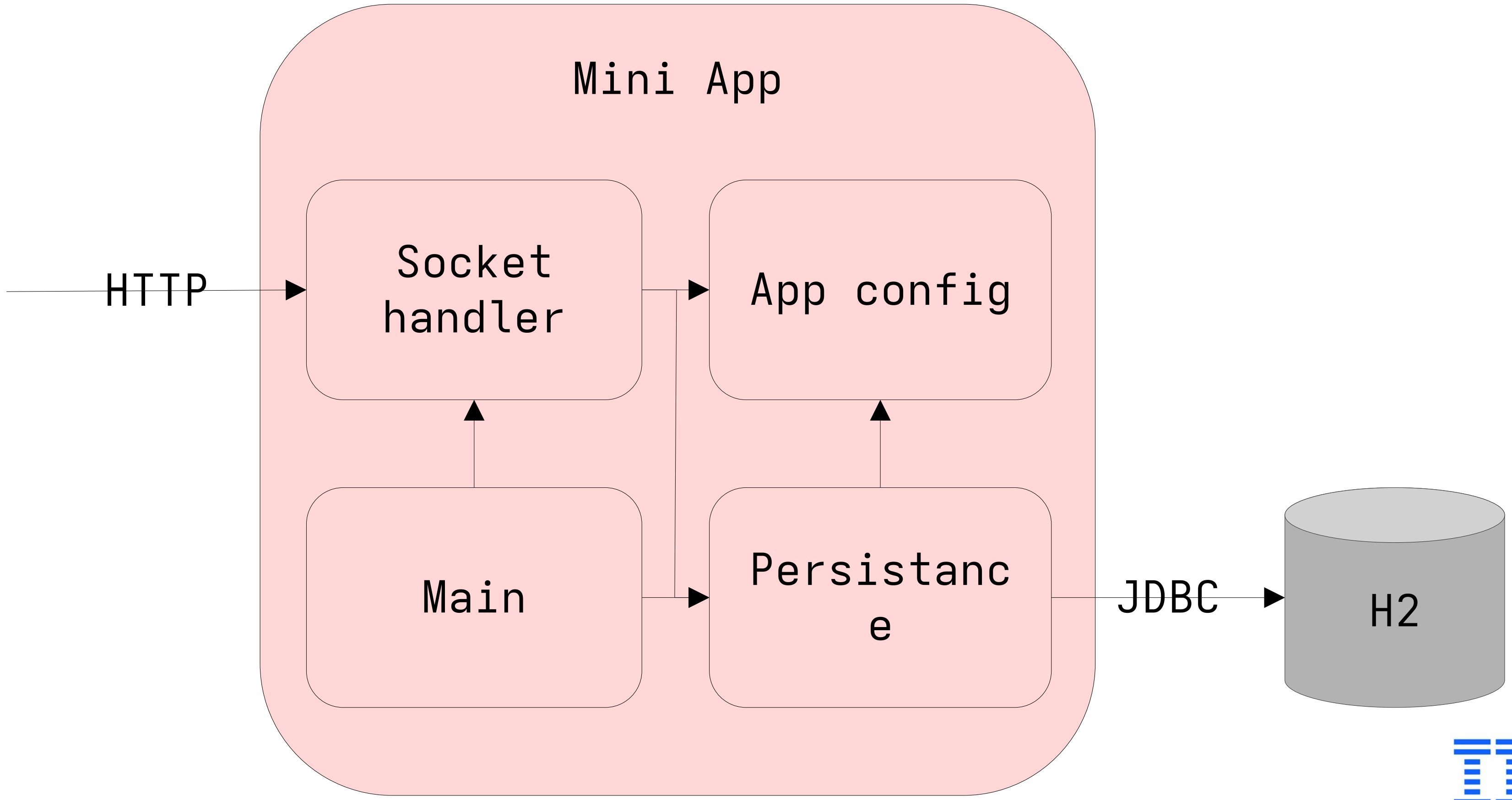
Paradigms & Instance flow control

- OOP
  - Encapsulation
  - Information hiding
  - Composition
  - Inheritance
  - Polymorphism
- Instance flow control
  - Dependency Injection
  - Inversion of Control

Maven

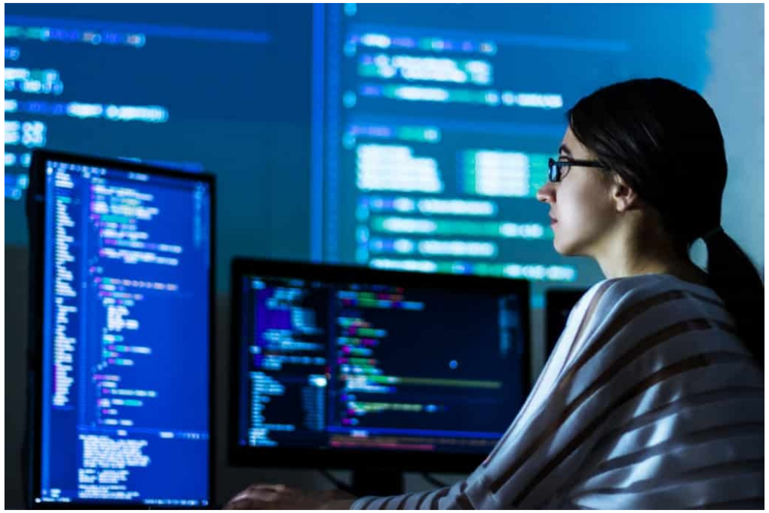
- Build automation tool for Java
- Uses `pom.xml` (Project Object Model)
- Project lifecycle phases
  - `compile`, `test`, `package`, `verify`, `install`, `deploy`
- Convention over configuration
- Multi module project support
- Also used to generate project structures for setup

Mini-app design



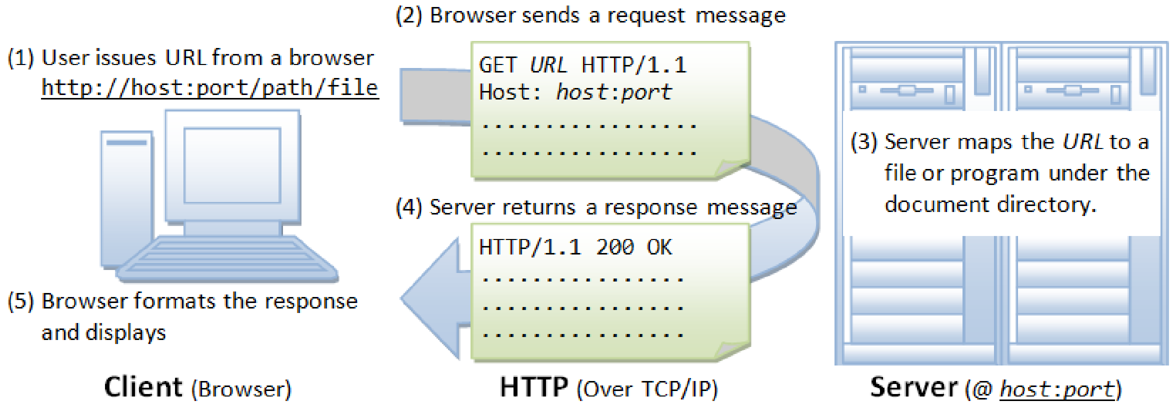
DEV: Create base application

- In VS Code remote session switch to `/home/student[n]/legacy` folder
- Checkout the git repo
  - `https://github.com/szabogabriel/FotS_2025_legacy`
- Init a new Maven application
  - CLI: `mvn init` …
  - VSCode: [F1] → Create maven app
  - Package: `com.ibm.sk.fots`
  - Application: `legacy`
- Create the main class
  - `App.java`
  - Print "`Hello, World`"

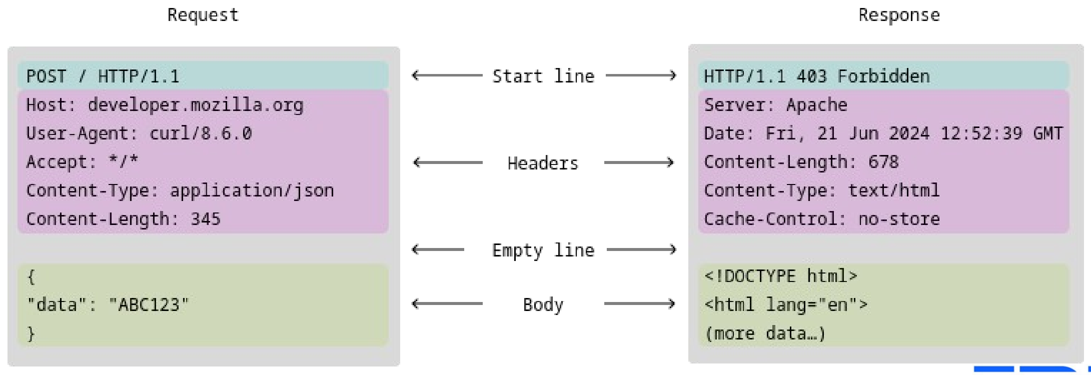


HTTP over sockets

- Communication via plaintext
- Request
  - first line
  - headers
  - (body)
- Response
  - status line
  - headers
  - (body)



(2) Browser sends a request message
(1) User issues URL from a browser http://host:port/path/file
GET URL HTTP/1.1
Host: host:port
(3) Server maps the URL to a file or program under the document directory.
(4) Server returns a response message
HTTP/1.1 200 OK
(5) Browser formats the response and displays
Client (Browser)
HTTP (Over TCP/IP)
Server (@ host:port)

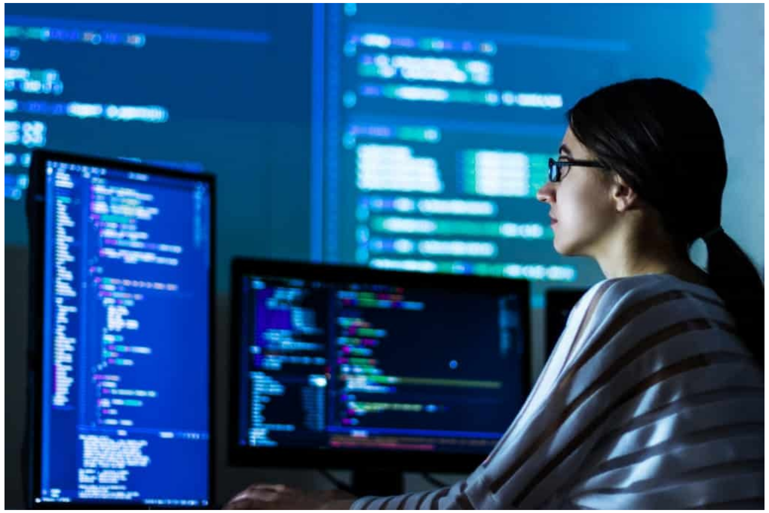


```
POST / HTTP/1.1
Host: developer.mozilla.org
User-Agent: curl/8.6.0
Accept: */*
Content-Type: application/json
Content-Length: 345

{
"data": "ABC123"
}
```

```
HTTP/1.1 403 Forbidden
Server: Apache
Date: Fri, 21 Jun 2024 12:52:39 GMT
Content-Length: 678
Content-Type: text/html
Cache-Control: no-store

<!DOCTYPE html>
<html lang="en">
(more data…)
```

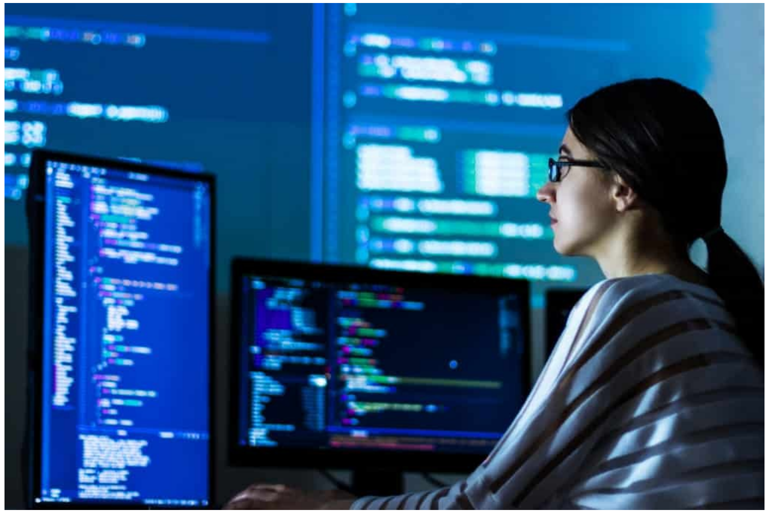
DEV: Server class

- Create a class “Server”
- Create a `ServerSocket` on a provided port
- Use `ServerSocket.accept` to create new communication Socket
- Read the data from the `Socket.getInputStream()`
- Write response to `Socket.getOutputStream()`
- Log data into `System.out`



DEV: enhance response

- Extend the response headers
  - content type
  - content length
- Extend response by the request method, path and remote IP
- Test with `curl`

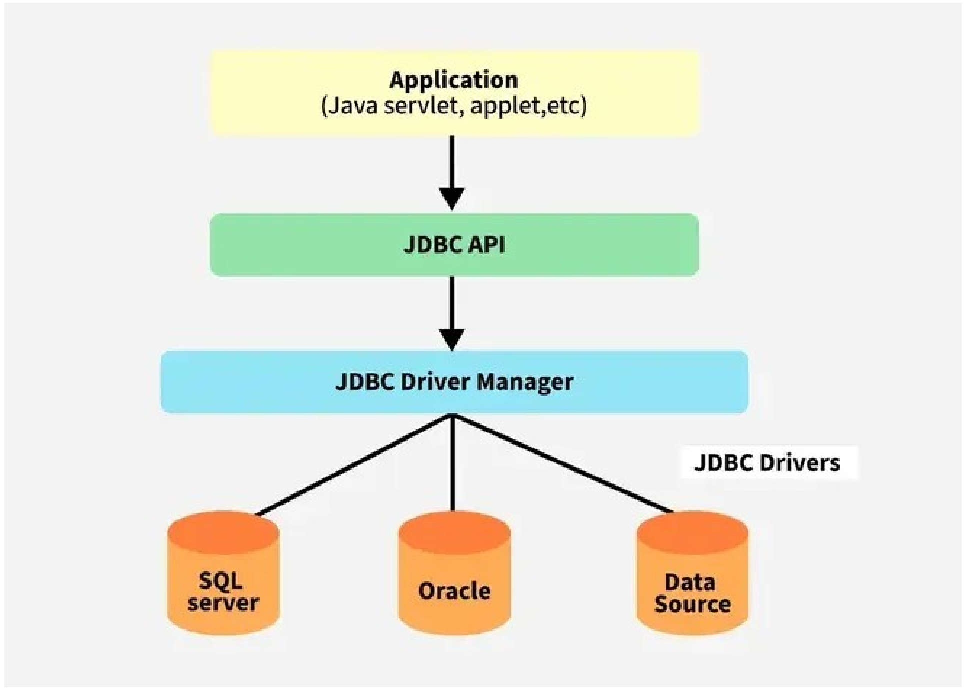


Pitfalls & quick fixes

- Double line breaks before body
  - `\r\n\r\n`
  - Read until empty line to consume every header
- Common ports
  - 80 - HTTP
  - 443 - HTTPS
  - 20, 21 - FTP
  - 22 - SSH
  - 23 - Telnet
  - 666 - Doom

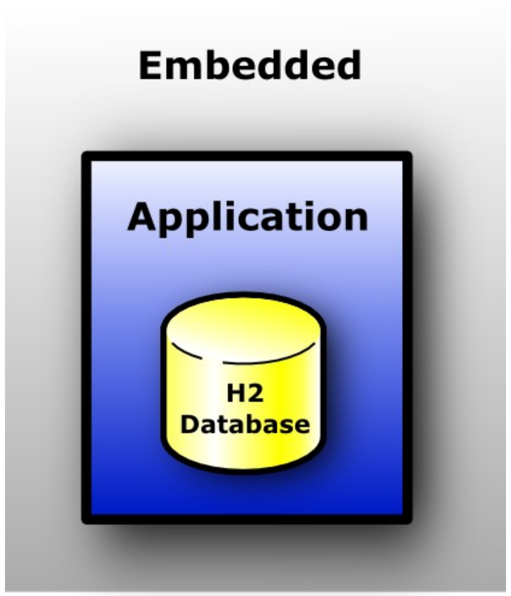
JDBC primer

- JDBC API - part of Java standard library
- Driver must be registered in class loader / classpath
  - `Class.forName(“driver name”);`
  - Per database (PostgreSQL, H2, Oracle, DB2 etc.)
  - External library (JAR) created by the DB provider
- Driver provides `Connection`
- `Connection` provides `PreparedStatement`
- Resources must be closed (explicitly or by try-with-resources)



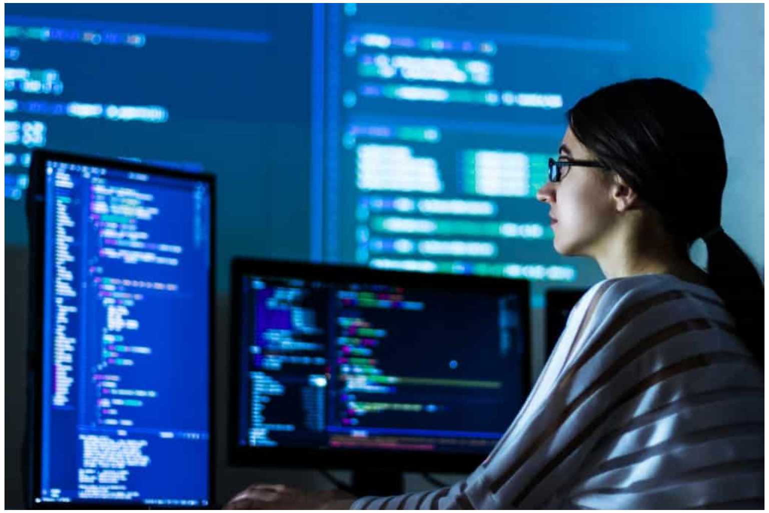
LOG_ENTRIES schema (H2)

- H2
  - embeddable DB
  - low memory and CPU footprint
  - mainly used in testing
  - support DB flavors (e.g. Oracle)
- Design table for log entries
  - ID, timestamp, content
  - create when not present upon startup



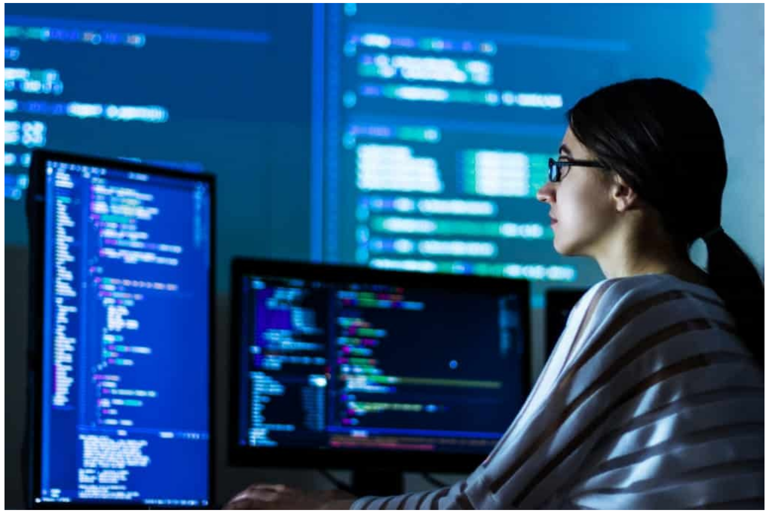
DEV: Create Database class

- Create a class `Database.java`
- Add H2 dependency to maven
- Load the JDBC driver
- Create a connection
- Init DB - LOG_ENTRIES
- Create a method for logging into the created table
- Start the H2 web console



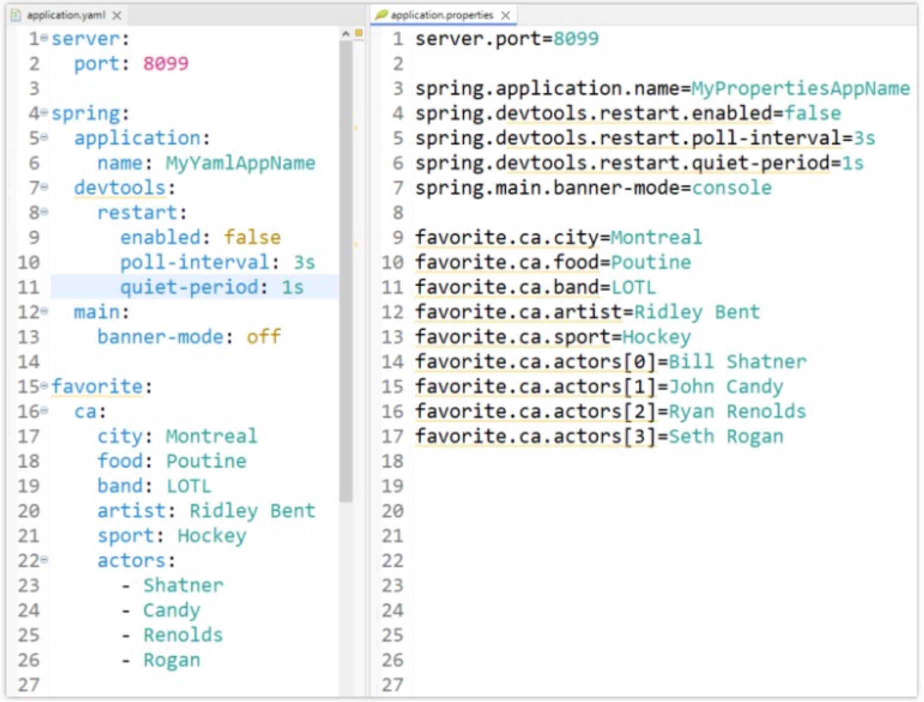
DEV: wire Server to Database

- Create a variable of type Database in Server
- Set the variable to an instance
- Swap the `System::out` calls with `Database::log` calls
- Test the application
- Check data via H2 console



Properties file format

- Standard way of configuring application
- Key-value format
- Supports comments (#, !)
- Structured data possible by dotting keys
- Near the JAR file (or packed)

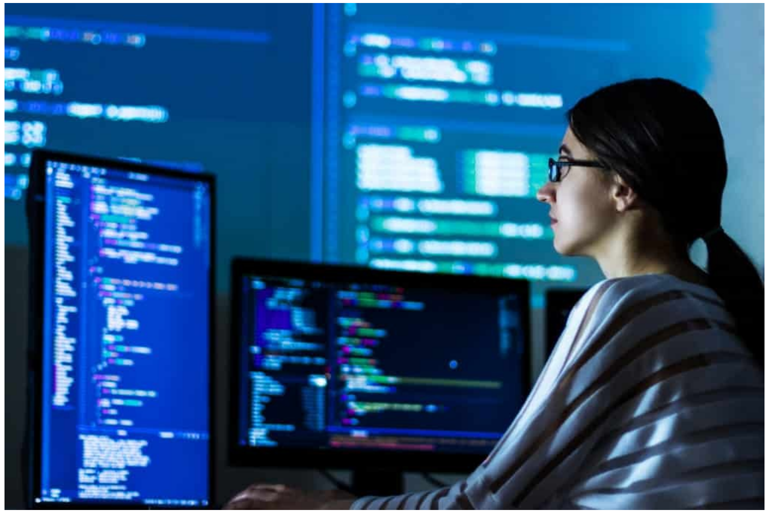


```yaml
server:
  port: 8099

spring:
  application:
    name: MyYamlAppName
  devtools:
    restart:
      enabled: false
      poll-interval: 3s
      quiet-period: 1s
  main:
    banner-mode: off

favorite:
  ca:
    city: Montreal
    food: Poutine
    band: LOTL
    artist: Ridley Bent
    sport: Hockey
    actors:
      - Shatner
      - Candy
      - Renolds
      - Rogan
```

```properties
server.port=8099

spring.application.name=MyPropertiesAppName
spring.devtools.restart.enabled=false
spring.devtools.restart.poll-interval=3s
spring.devtools.restart.quiet-period=1s
spring.main.banner-mode=console

favorite.ca.city=Montreal
favorite.ca.food=Poutine
favorite.ca.band=LOTL
favorite.ca.artist=Ridley Bent
favorite.ca.sport=Hockey
favorite.ca.actors[0]=Bill Shatner
favorite.ca.actors[1]=John Candy
favorite.ca.actors[2]=Ryan Renolds
favorite.ca.actors[3]=Seth Rogan
```

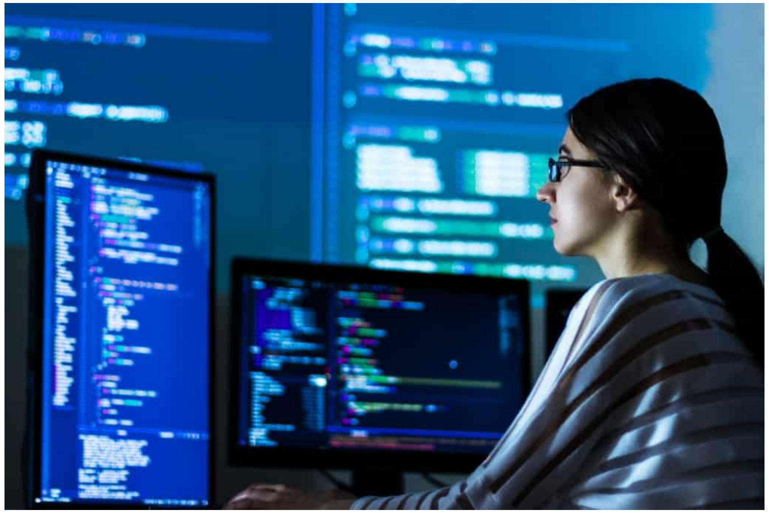
DEV: Config loader

- Create a Config enum
- Create an enum value for every property in project
- Set a default value
- Create Config.getValue. Return
  - value from the environment variable (`System.getEnv()`)
  - default one



DEV: run with custom config

- Create `app.properties` file
- Create an entry in the property file for every enum instance
- Load the property file via `Properties.load`
- Update property loader – make values in the property file default



End to end test

- Create a few requests
- Show the H2 console
- Change property for the port by increasing it by 50
- Re-run the application

Lessons lerned and why Spring helps

- Wrap up
  - Java library is huge
  - Java library is complex
  - Sometimes PitA (errors, typos, effort)
- DI / IoC – gives us back some control
- Next → We Boot up Spring

Wrap-up, homework and buffer

- Homework (Optional)
  - Add a request path & header User-Agent to DB
  - Add a log rotation to → new file DB per day
  - GET /health endpoint on the server
  - Create a short README about the app
- Discussion

Thank you

Optional 1: Thread executor

- Create a ThreadPoolExecutor in the Server class
- Use `ThreadPoolExecutor::execute` instead of `Thread::start`

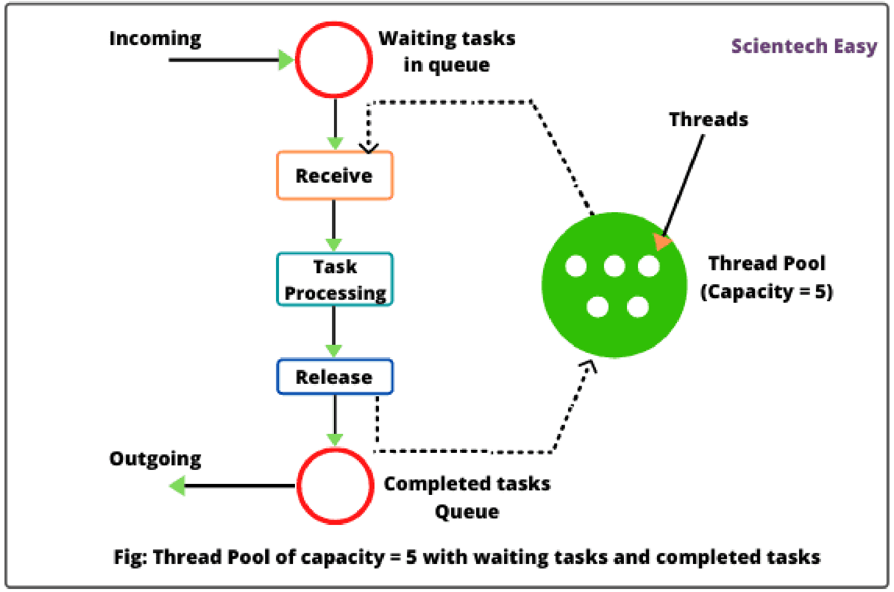
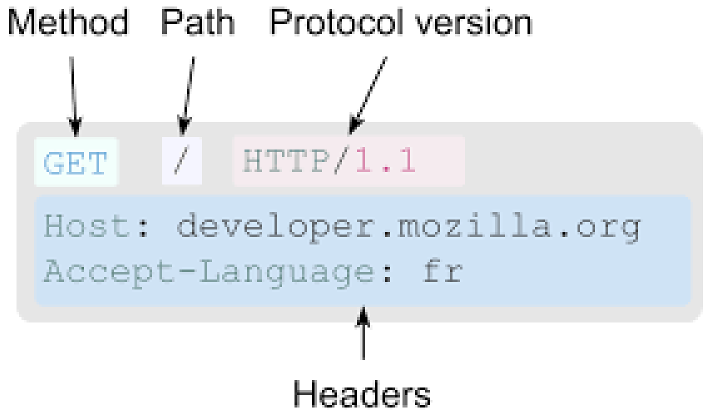


Fig: Thread Pool of capacity = 5 with waiting tasks and completed tasks

Optional 2: Improve request parsing

- Recognize HTTP method
  - GET / HEAD → no body
- Set content length properly
- Handle request path
  - Find the path from header
  - Store in separate DB table



```
GET   /   HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

Optional 3: Validation and logging

- Trim long entries to fit into DB column
- Sanitize non printable characters
- Ignore suspicious headers
- Swap `System.out` to custom logger class

Welcome, logistics & setup (30min)

- Logistics / setup: 30m
- Git: 37m
- Java recap: 28m
- Sockets / Hello: 28m
- JDBC/H2: 23m
- Config: 9m
- Wrap: 13m
- → cca 168m w 12m buffer