Session 1 Handout — Git, curl & JDBC (1-pager)

Keep this nearby while coding. Works on the remote server provided in class.

---

0) Setup

Server information

IP: 91.98.156.163

`bash`

`ssh username@91.98.156.163`

Account information

Username

Password

Basic Linux commands

```
# Directory / file navigation and manipulation
cd dirname
mkdir dirname
cp file.txt target.txt
rm file.txt

# Looking for data in files
grep -e 'pattern' file.txt
less file.txt
cat file.txt

# Operations on strings
tr 'a' 'b'
sed 's/pattern/goal/g'

# Finding files
find . -type f -name 'name-pattern'
```

---

1) Git — core workflow

Configure (once):

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

Clone to current folder & branch:

```
git clone <REPO_URL> .
cd <repo>
git checkout -b feature/<branch_name>
```

Status → add → commit → push:

```
git status
git add <files>     # or: git add -A
git commit -m "feat: add greeting"
git push -u origin HEAD
```

Sync & resolve conflicts:

```
git pull --rebase    # get latest main into your branch
# open conflicted files in VS Code, fix, then:
git add <fixed-files>
git rebase --continue # or: git commit
```

Merge (via PR) or locally:

```
git checkout main && git pull
git merge feature/<firstname>
```

Undo tips:

```
git restore --staged <file>   # unstage
git checkout -- <file>         # discard local change
```

---

2) curl — quick HTTP checks

Simple GET (verbose):

```
curl -v http://localhost:8080/hello
```

Send a header & show response headers only:

```
curl -s -D- -o /dev/null -H "X-User: Alice" http://localhost:8080/hello
```

POST JSON:

```
curl -v -H "Content-Type: application/json" \
     -d '{"title":"First task","status":"TODO"}' \
     http://localhost:8080/api/tasks
```

---

3) H2 + JDBC — minimal snippets

Maven dependency (add to `pom.xml`):

```
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <version>2.2.224</version>
</dependency>
```

Register JDBC driver in classpath

```
Class.forName("org.h2.Driver");
```

JDBC URLs (dev-friendly):

```
# File DB (persists on disk)
jdbc:h2:file:./data/appdb;DB_CLOSE_DELAY=-1;AUTO_SERVER=TRUE
# In-memory (dies with JVM)
jdbc:h2:mem:test;DB_CLOSE_DELAY=-1
```

Create table if missing (LOG_ENTRIES):

```
CREATE TABLE IF NOT EXISTS LOG_ENTRIES (
  ID BIGINT AUTO_INCREMENT PRIMARY KEY,
  TS TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONTENT VARCHAR(4000) NOT NULL
);
```

Java helper (open DB, insert log):

```
import java.sql.*;

class Database implements AutoCloseable {
  private final Connection conn;
  Database(String url) {
    try {
      Class.forName("org.h2.Driver");
      this.conn = DriverManager.getConnection(url, "sa", "");
      try (var st = conn.createStatement()) {
        st.execute("""
          CREATE TABLE IF NOT EXISTS LOG_ENTRIES(
            ID BIGINT AUTO_INCREMENT PRIMARY KEY,
            TS TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
            CONTENT VARCHAR(4000) NOT NULL)
        """);
      }
    } catch (Exception e) { throw new RuntimeException(e); }
  }
  long log(String content) {
    try (var ps = conn.prepareStatement(
          "INSERT INTO LOG_ENTRIES(CONTENT) VALUES (?)",
          Statement.RETURN_GENERATED_KEYS)) {
      ps.setString(1, content);
      ps.executeUpdate();
      try (var rs = ps.getGeneratedKeys()) { return rs.next() ? rs.getLong(1) : -1L; }
    } catch (SQLException e) { throw new RuntimeException(e); }
  }
  public void close() {
    try { conn.close(); } catch (Exception ignored) {}
  }
}
```

(Optional) H2 web console in code:

```
// http://localhost:8082/
org.h2.tools.Server.createWebServer("-web","-webPort","8082","-webDaemon").start();
```

---

4) Sockets and communication

Listen to incomming connection

```
import java.io.*;
import java.net.*;

public class Server() {

  public Server() {
    ServerSocket serverSocket = new ServerSocket(8080);
    while (true) {
      try {
        Socket socket = serverSocket.accept(); // Blocking call
        handle(socket);
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
  }

  private void handle(Socket socket) {
    new Thread(() -> {
      try {
        InputStream in = socket.getInputStream();
        OutputStream out = socket.getOutputStream();
        String request = readInputStream(in);
        out.write(createResponse(request));
        out.flush();
        socket.close();
      }
    }).start();
  }

  private String readInputStream(InputStream in) {
    //TODO
  }

  private String createResponse(OutputStream out) {
    //TODO
  }
}
```

5) Config file + wiring

app.properties

```
server.port=8080
db.url=jdbc:h2:file:./data/appdb;DB_CLOSE_DELAY=-1;AUTO_SERVER=TRUE
```

Load properties:

```
import java.util.*;
import java.nio.file.*;

public enum Config {
  PORT("server.port","8080"),
```

```
  DB_USERNAME("db.username","admin"),
  DB_PASSWORD("db.password","admin"),
  ;

  private final String key;
  private final String defaultValue;

  Config(String key, String defaultValue) {
    this.key = key;
    this.defaultValue = defaultValue;
  }

  public String value() {
    String envValue = System.getenv(key.toUpperCase());
    if (envValue != null) {
      return envValue;
    }

    Properties p = new Properties();
    try (var in = Files.newInputStream(Path.of("app.properties"))) {
      p.load(in);
    }

    return p.getProperty(key, defaultValue);
  }
}
```

---

6) Tools

Maven:

```
# Generate new project
mvn archetype:generate -DgroupId=com.example.app -DartifactId=my-app -DarchetypeArtifactId=maven-archety

# Build project without tests
mvn -q -DskipTests package
```

Run plain Java app

```
java -cp target/classes:~/.m2/repository/com/h2database/h2/2.2.224/h2-2.2.224.jar your.pkg.Main
```

(Windows `-cp` uses `;` instead of `:`)

cURL smoke test:

```
curl -v http://localhost:8080/hello
```

Quick tips

- Keep commits small; write meaningful messages.
- Always read until an empty line when parsing HTTP headers over sockets.
- Prefer file-based H2 during class (`jdbc:h2:file:...`) so data survives restarts.
- If port 8080 is busy, change `server.port` in `app.properties`.