Java + Hibernate ORM (Object Relational Mapping): Database Lab

A hands-on lab for connecting Java applications with relational databases using Hibernate ORM framework.

โดย **Sarawoot Kongyoung**

Lab Title

Java + Hibernate ORM

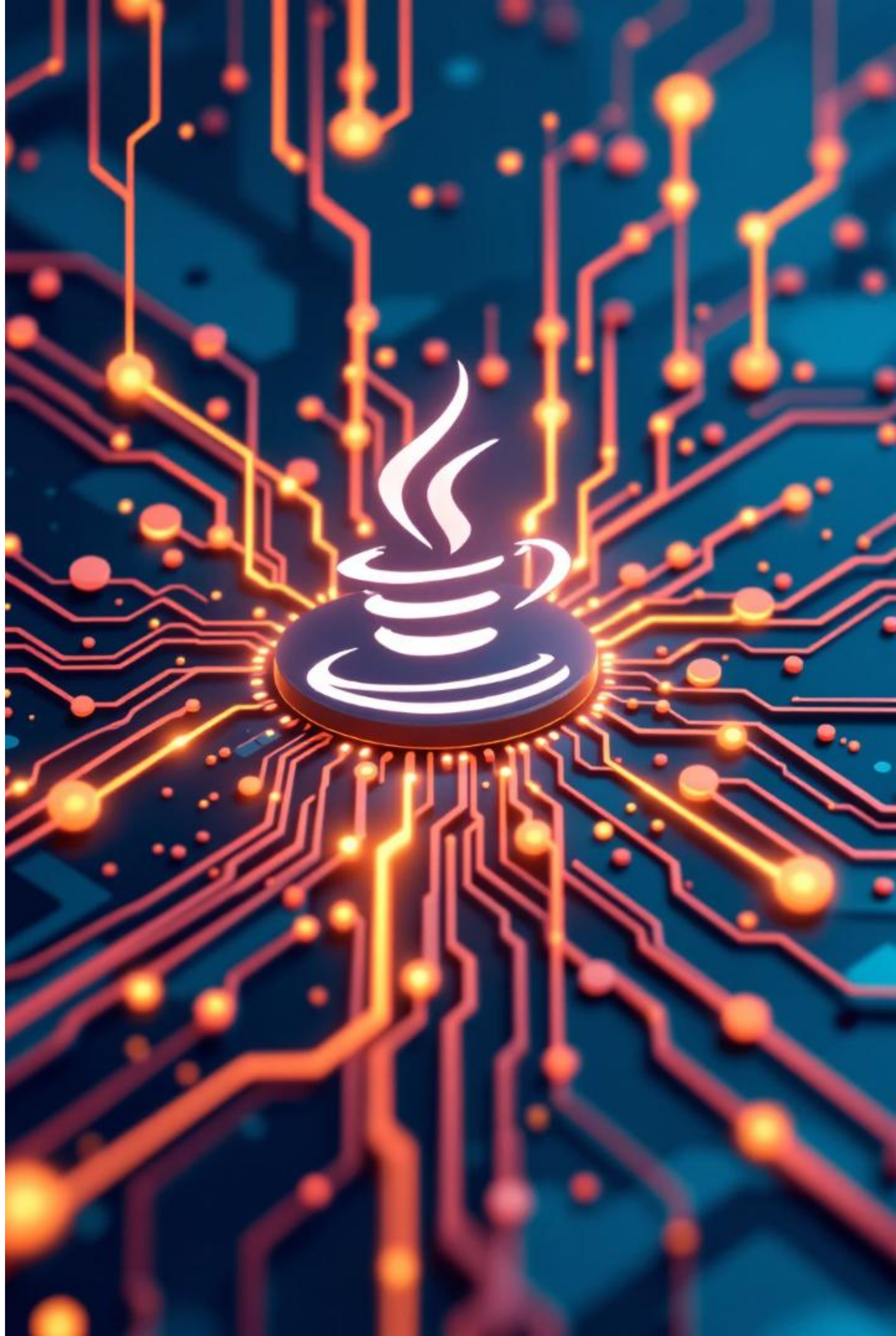
Connecting Java with Relational Databases

Practical Focus

Hands-on database integration

ORM Approach

Object-Relational Mapping techniques



What is Hibernate?

- Java-based ORM framework
- Maps Java classes to database tables
- Simplifies database operations

Why Use ORM?

Avoid Boilerplate

Eliminates repetitive JDBC code

Object Focus

Work with Java objects, not SQL

Schema Mapping

Automatic table-class conversion

Relationships

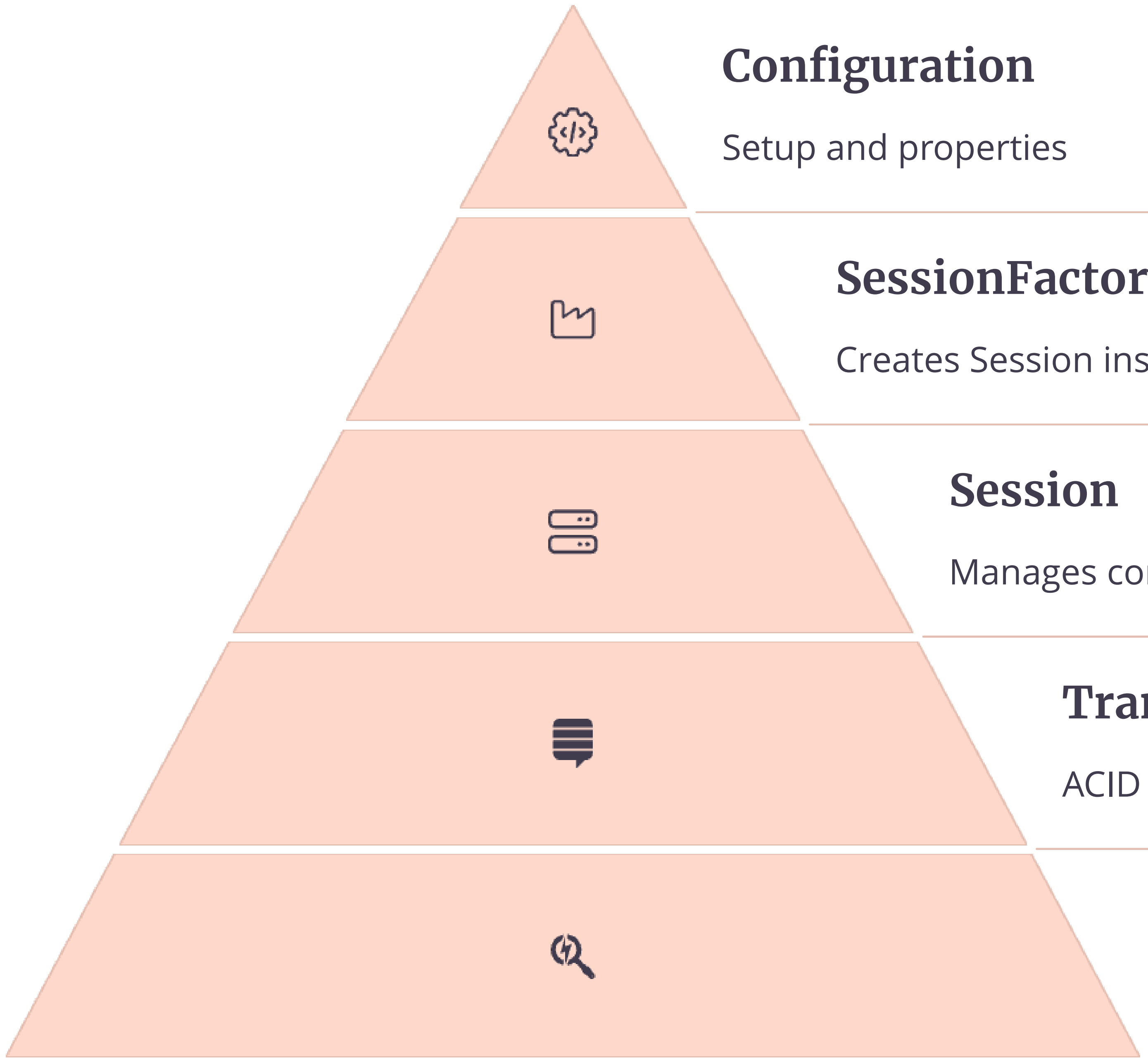
Supports 1:1, 1:N connections

```
import java.sql.*;

public class PostgresJDBCExample {

  public static void main(String[] args) {
    String url = "jdbc:postgresql://localhost:5432/your_database";
    String user = "your_username";
    String password = "your_password";

    try (Connection conn = DriverManager.getConnection(url, user, password)) {
        System.out.println("✅ Connected to PostgreSQL!");

      // Insert an employee
      String insertSQL = "INSERT INTO employee (name, department, position, start_date) VALUES (?, ?, ?, ?)";
      try (PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {
        pstmt.setString(1, "Nok");
        pstmt.setString(2, "IT");
        pstmt.setString(3, "Developer");
        pstmt.setDate(4, Date.valueOf("2025-04-01"));
        pstmt.executeUpdate();
          System.out.println("✅ Employee inserted");
      }

      // Query all employees
      String querySQL = "SELECT * FROM employee";
      try (Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(querySQL)) {
        while (rs.next()) {
            System.out.printf("👤 ID: %d | Name: %s | Dept: %s | Pos: %s | Start: %s%n",
            rs.getInt("emp_id"),
            rs.getString("name"),
            rs.getString("department"),
            rs.getString("position"),
            rs.getDate("start_date"));
        }
      }

    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

🧠 Summary: JDBC vs Hibernate

Feature	JDBC	Hibernate ORM
Code Level	Low-level SQL	High-level OOP
Mapping	Manual (tables ↔ fields)	Automatic (via annotations/XML)
Query Language	SQL	HQL (object-oriented SQL)
Relationships	Manual joins	@OneToMany, @ManyToOne, etc.
Transactions	Explicit (commit/rollback)	Simplified with Session/Transaction

Hibernate Architecture



Configuration

Setup and properties

SessionFactory

Creates Session instances

Session

Manages connections

Transaction

ACID operations

HQL

Query language

Setup Requirements

JDK 17+

Latest Java Development Kit

IDE

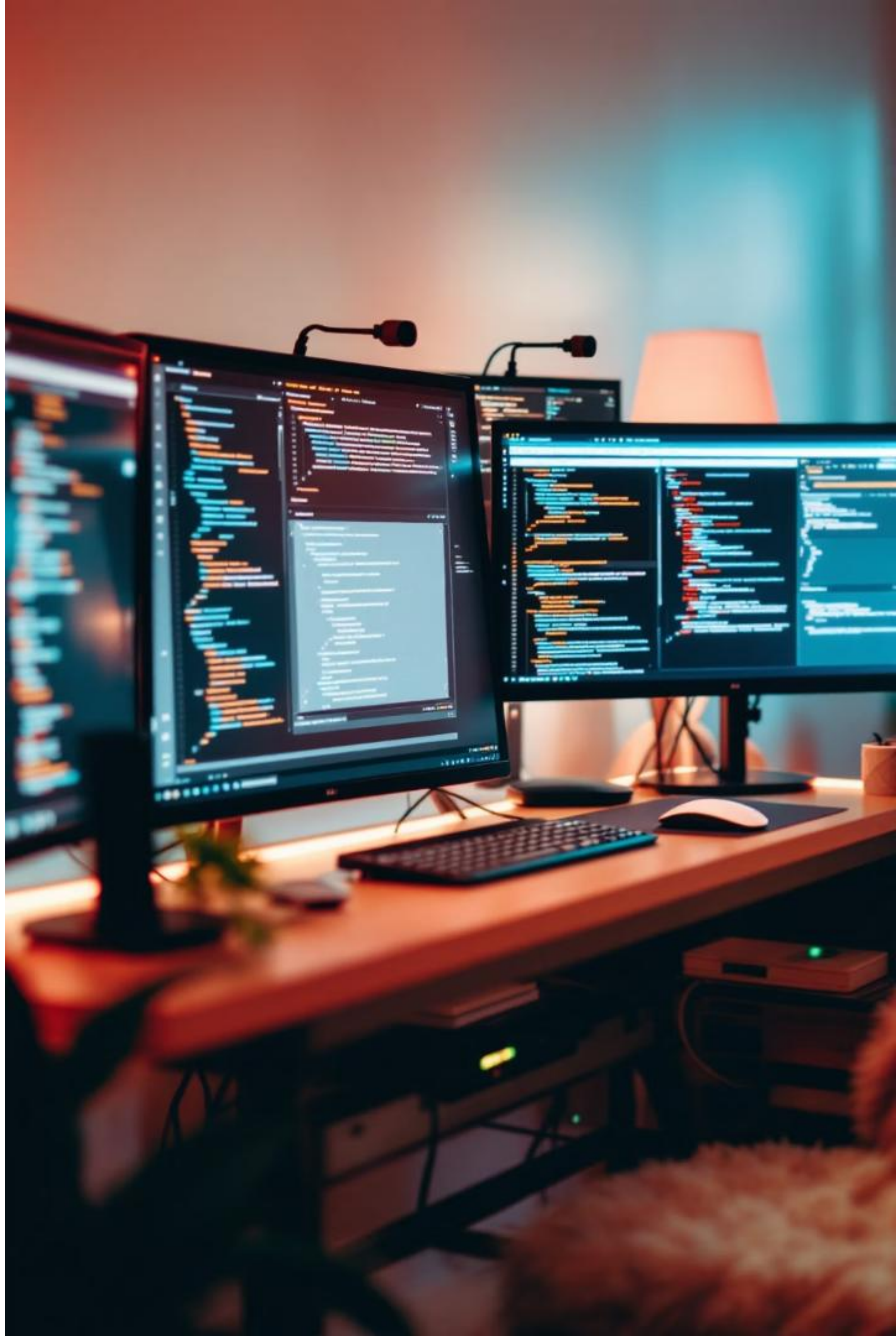
Eclipse or VS Code

Maven

Dependency management

Database

MySQL or SQLite



Install Eclipse

Download

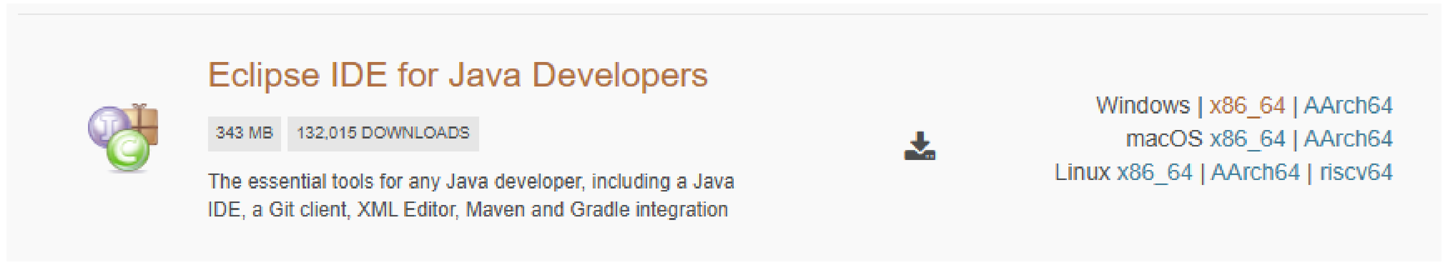
Get Eclipse from eclipse.org/downloads

Select Package

Choose "Eclipse IDE for Java Developers"

Install

Extract and launch the application


Eclipse IDE for Java Developers

343 MB 132,015 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration

Windows | x86_64 | AArch64
macOS x86_64 | AArch64
Linux x86_64 | AArch64 | riscv64

Install VS Code (Optional)

Download VS Code

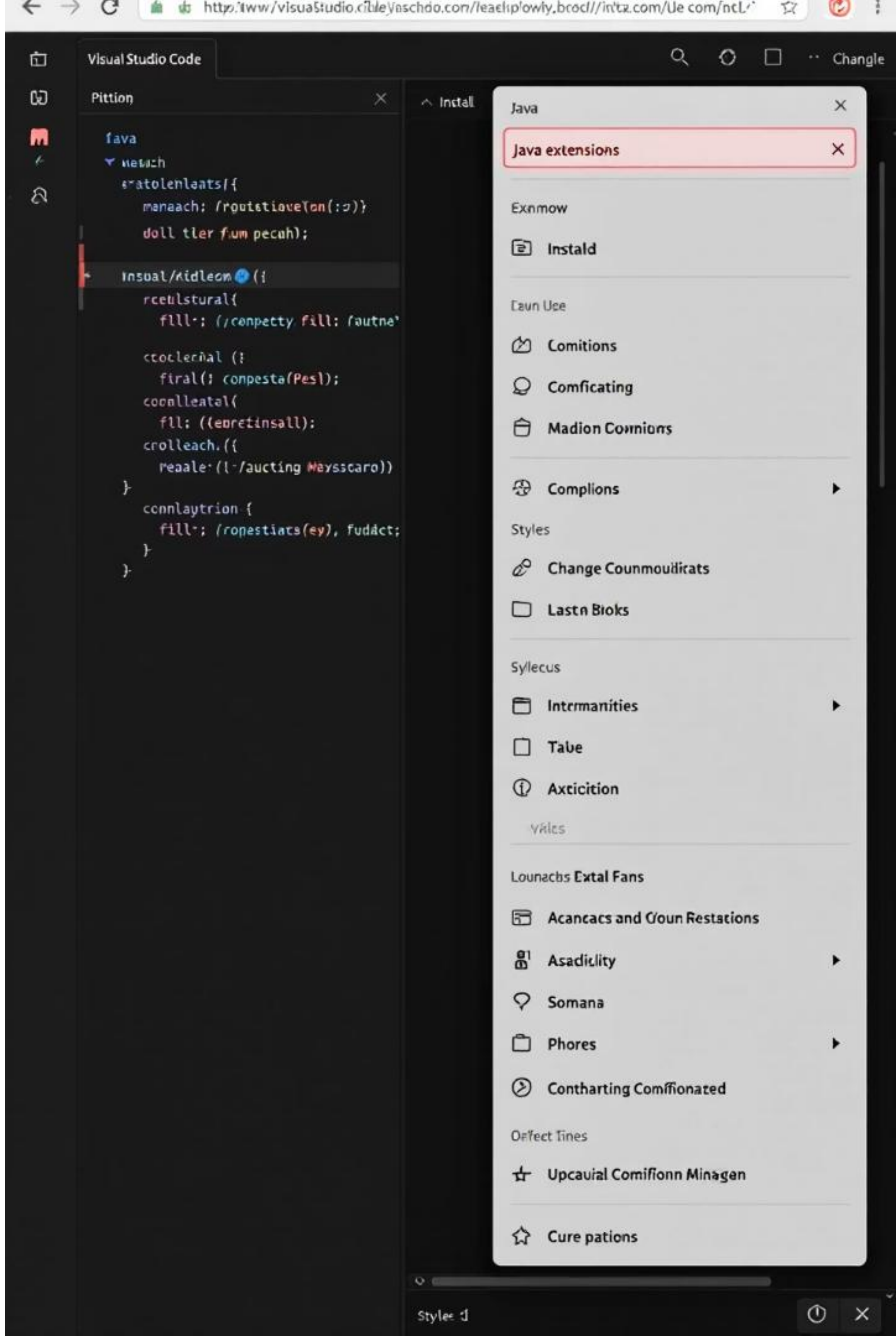
From code.visualstudio.com

Install Extensions

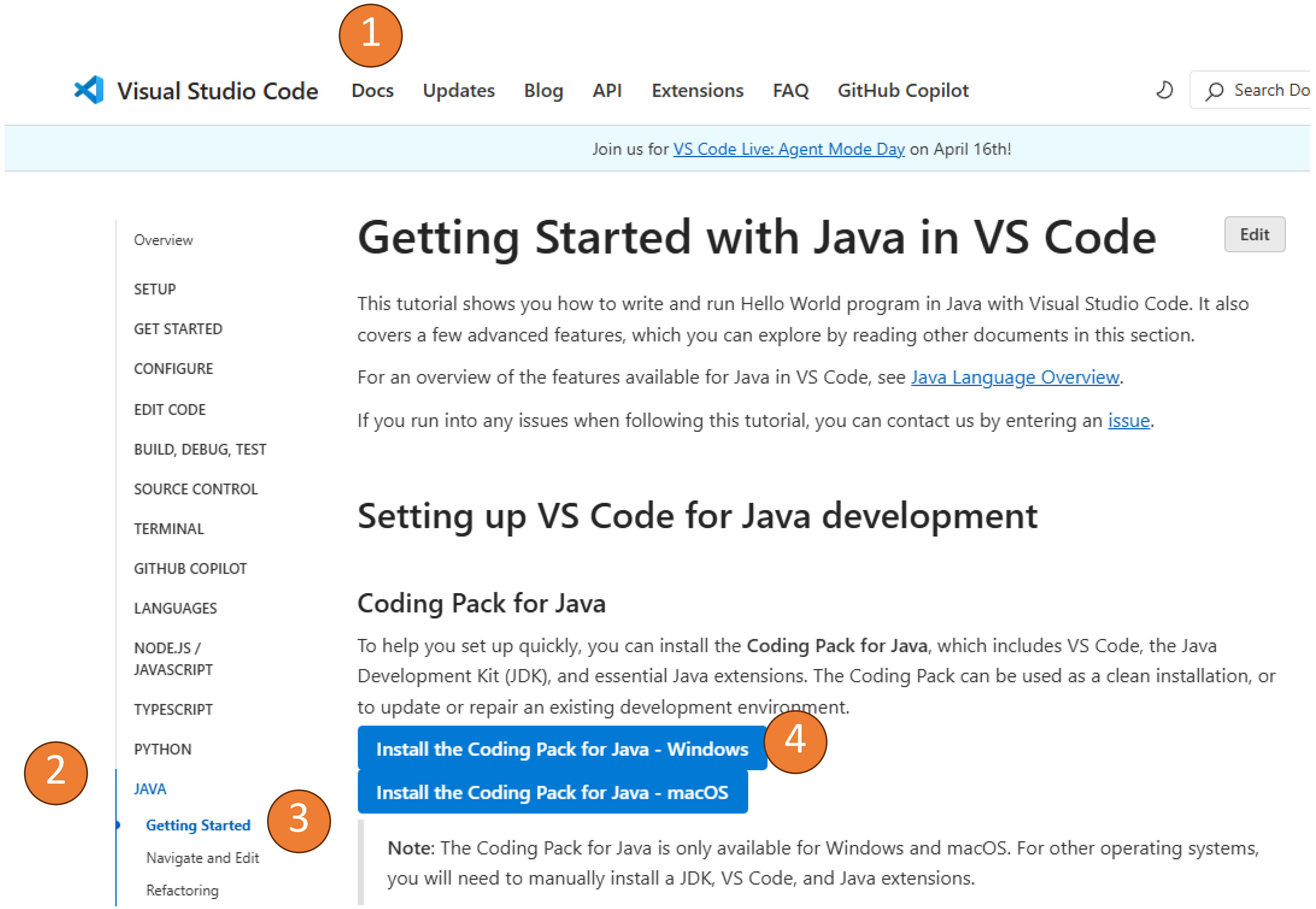
Add Java Extension Pack

Configure

Set up Java path in settings



```
fava
▾ netach
  eratolenlaats({
    manaach: (routstiove(on(:o))}
    doll tler fum pecoh);

  insual/kidleon ({
    rceulstural{
      flll·: (/conpecty fill: fautne'

    ctoclerhal (!
      firal(: conpesta(Pesl);
    coonlleatal(
      fll: ((eoretinsall);
    crolleach.({
      reaale·(l-/aucting Meysscaro))
  }
    connlaytrion {
      fill·; (ropestiats(ey), fuddct;
    }
  }
```


Visual Studio Code Docs Updates Blog API Extensions FAQ GitHub Copilot Search Do

Join us for VS Code Live: Agent Mode Day on April 16th!

Getting Started with Java in VS Code

This tutorial shows you how to write and run Hello World program in Java with Visual Studio Code. It also covers a few advanced features, which you can explore by reading other documents in this section.

For an overview of the features available for Java in VS Code, see Java Language Overview.

If you run into any issues when following this tutorial, you can contact us by entering an issue.

Setting up VS Code for Java development

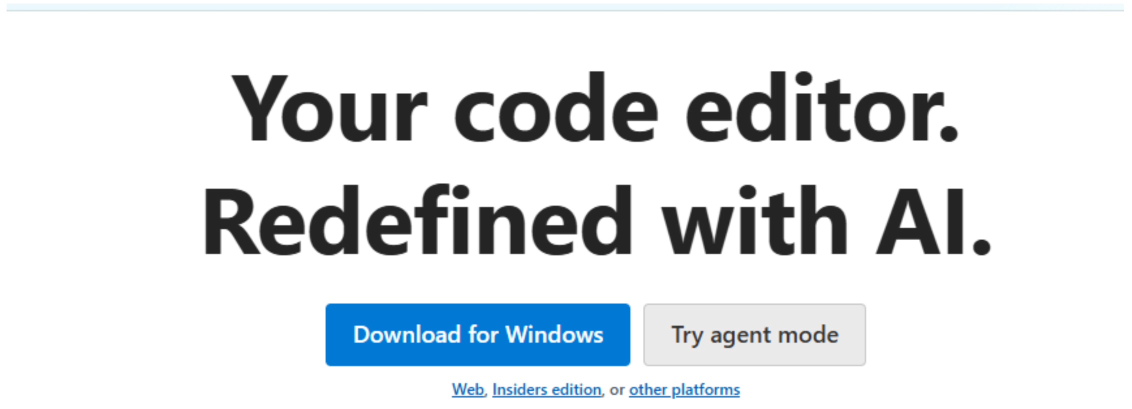
Coding Pack for Java

To help you set up quickly, you can install the **Coding Pack for Java**, which includes VS Code, the Java Development Kit (JDK), and essential Java extensions. The Coding Pack can be used as a clean installation, or to update or repair an existing development environment.

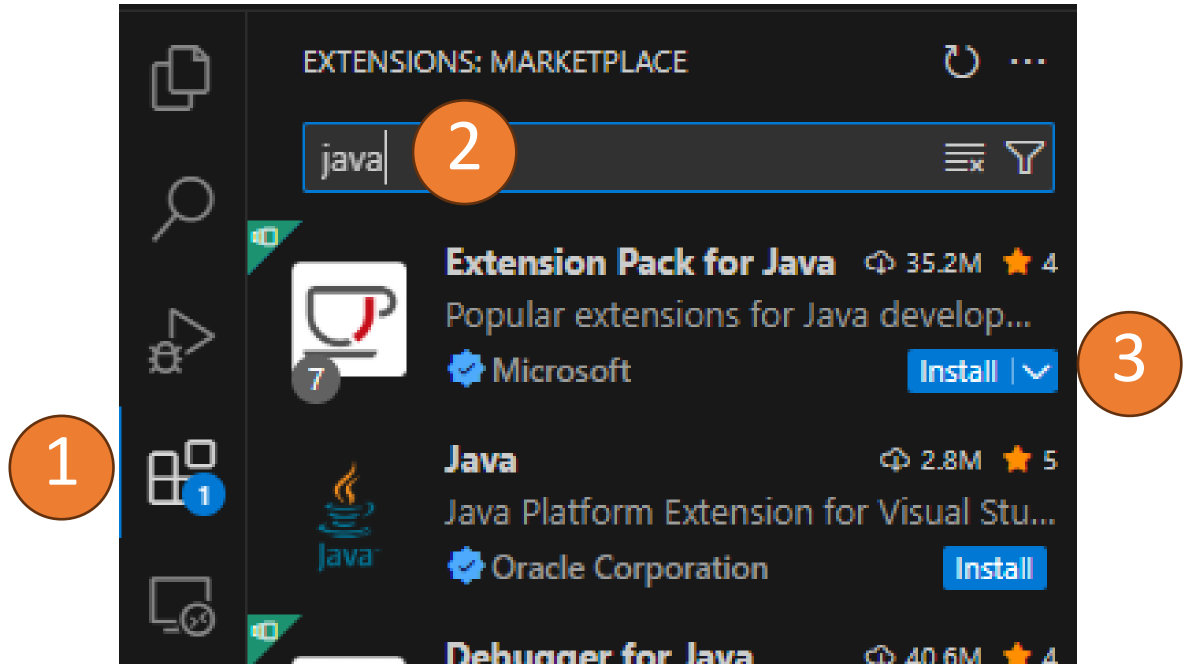
Install the Coding Pack for Java - Windows

Install the Coding Pack for Java - macOS

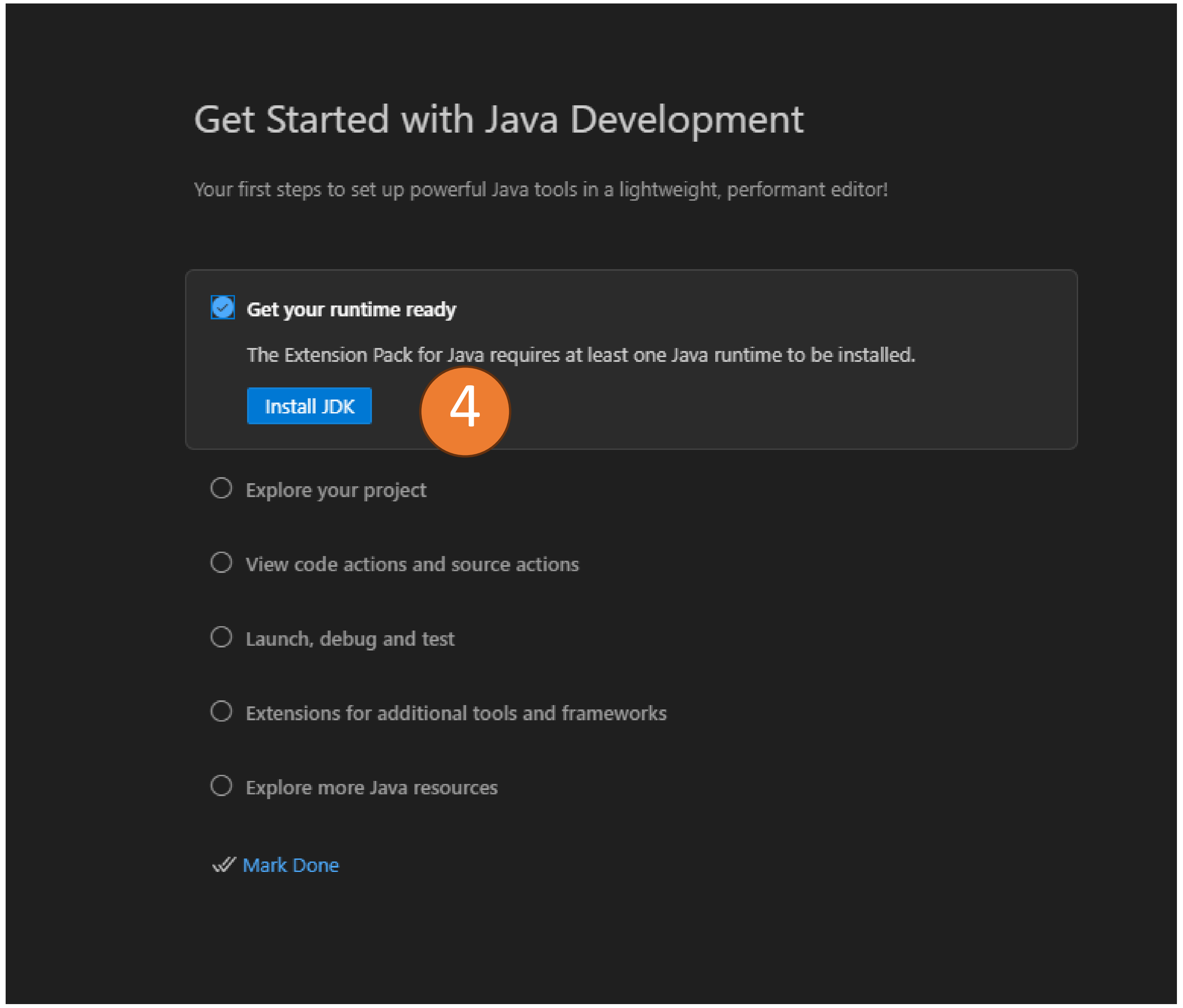
Note: The Coding Pack for Java is only available for Windows and macOS. For other operating systems, you will need to manually install a JDK, VS Code, and Java extensions.



Your code editor.
Redefined with AI.



EXTENSIONS: MARKETPLACE
java
Extension Pack for Java 35.2M 4
Popular extensions for Java develop...
Microsoft
Install
Java 2.8M 5
Java Platform Extension for Visual Stu...
Oracle Corporation
Install
Debugger for Java 40.6M 4
1
2
3



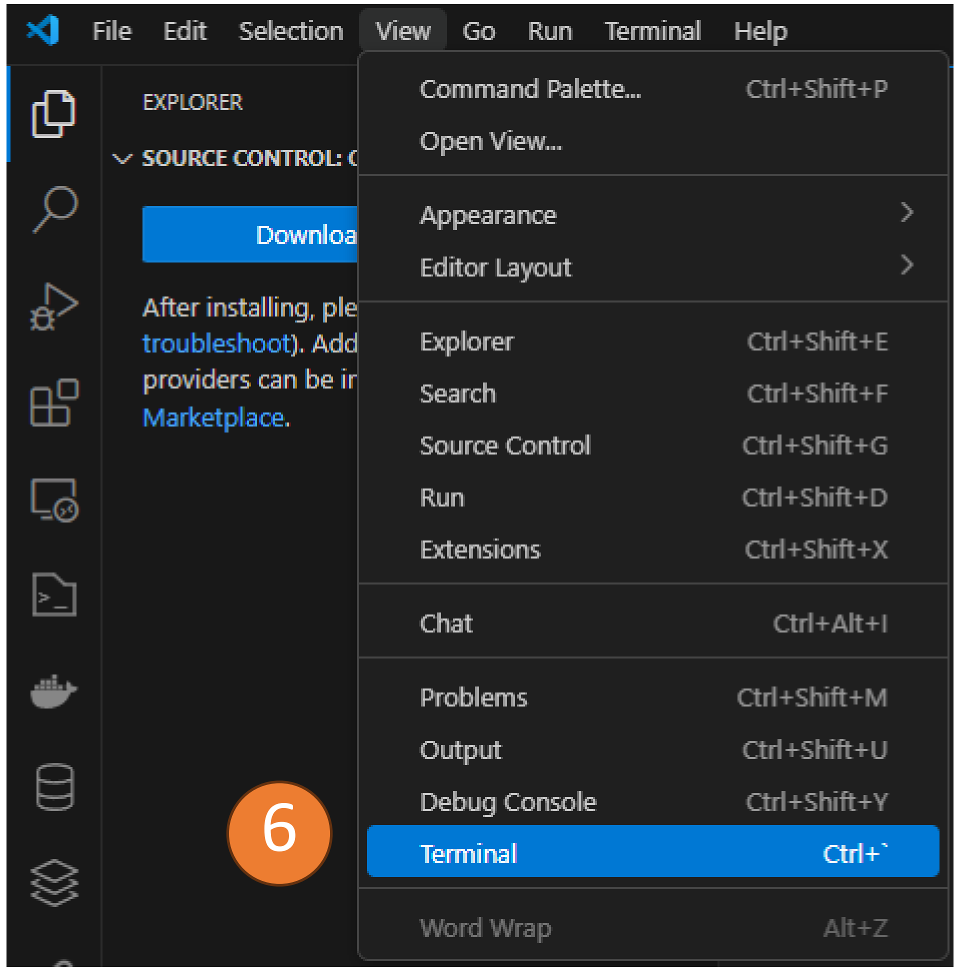
Get Started with Java Development

Your first steps to set up powerful Java tools in a lightweight, performant editor!

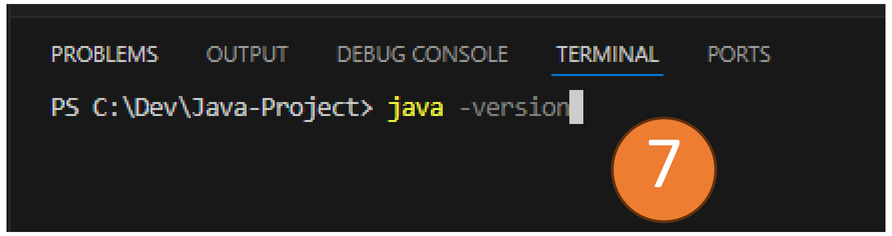
- Get your runtime ready

  The Extension Pack for Java requires at least one Java runtime to be installed.
- Explore your project
- View code actions and source actions
- Launch, debug and test
- Extensions for additional tools and frameworks
- Explore more Java resources

5



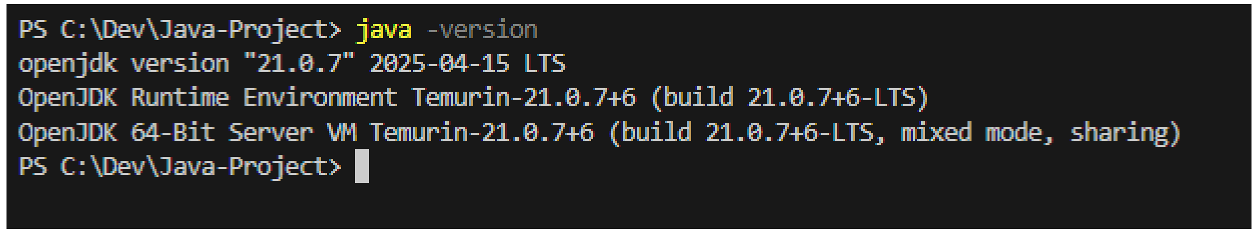
6



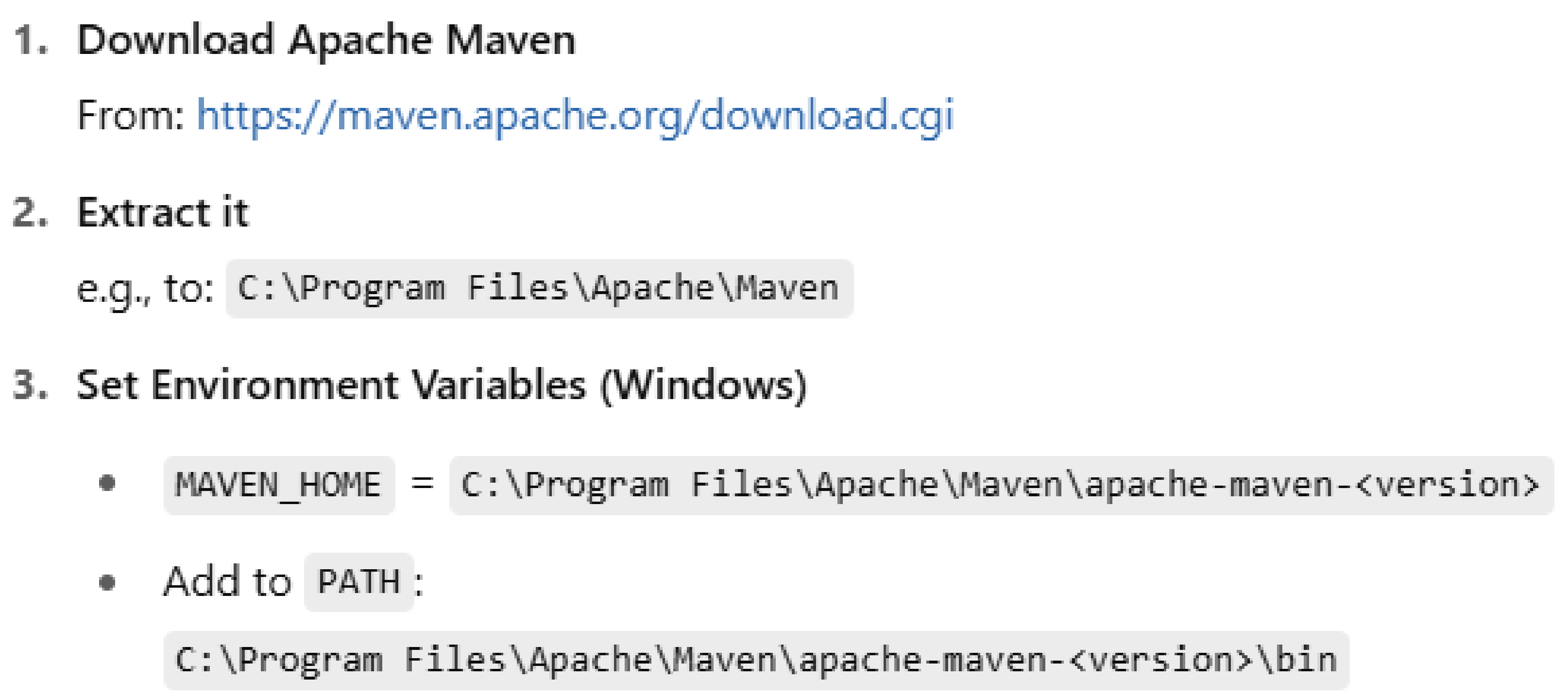
```
PS C:\Dev\Java-Project> java -version
```

7

8



```
PS C:\Dev\Java-Project> java -version
openjdk version "21.0.7" 2025-04-15 LTS
OpenJDK Runtime Environment Temurin-21.0.7+6 (build 21.0.7+6-LTS)
OpenJDK 64-Bit Server VM Temurin-21.0.7+6 (build 21.0.7+6-LTS, mixed mode, sharing)
PS C:\Dev\Java-Project>
```



1. **Download Apache Maven**

   From: https://maven.apache.org/download.cgi

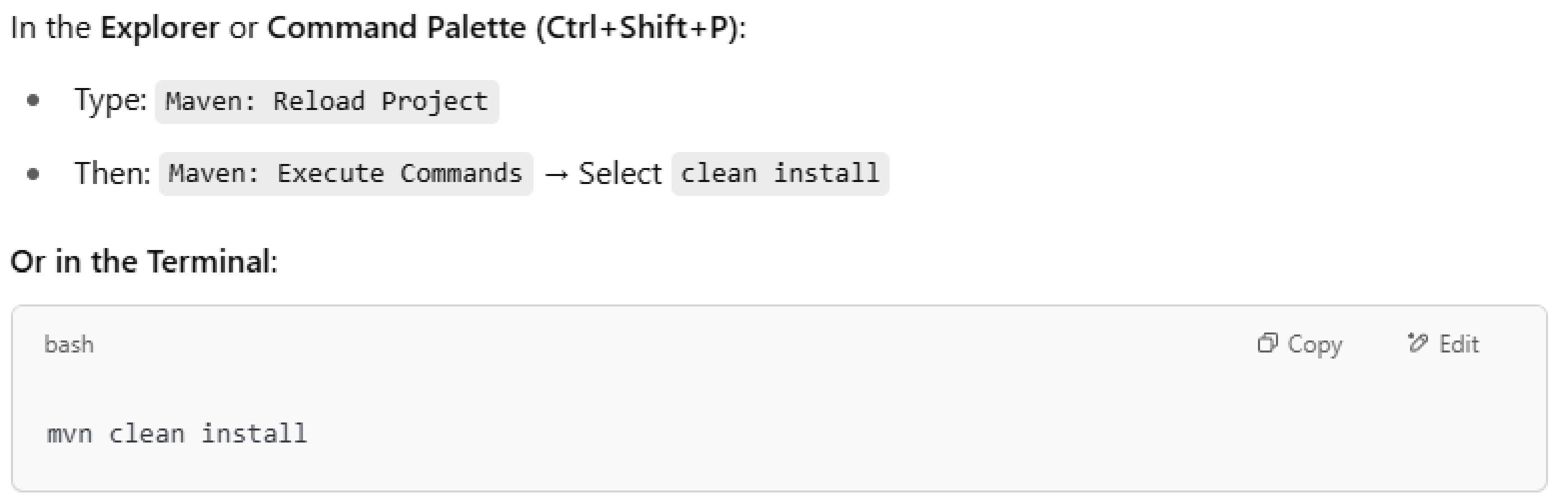
2. **Extract it**

   e.g., to: `C:\Program Files\Apache\Maven`

3. **Set Environment Variables (Windows)**

   - `MAVEN_HOME` = `C:\Program Files\Apache\Maven\apache-maven-<version>`
   - Add to `PATH`:

     `C:\Program Files\Apache\Maven\apache-maven-<version>\bin`

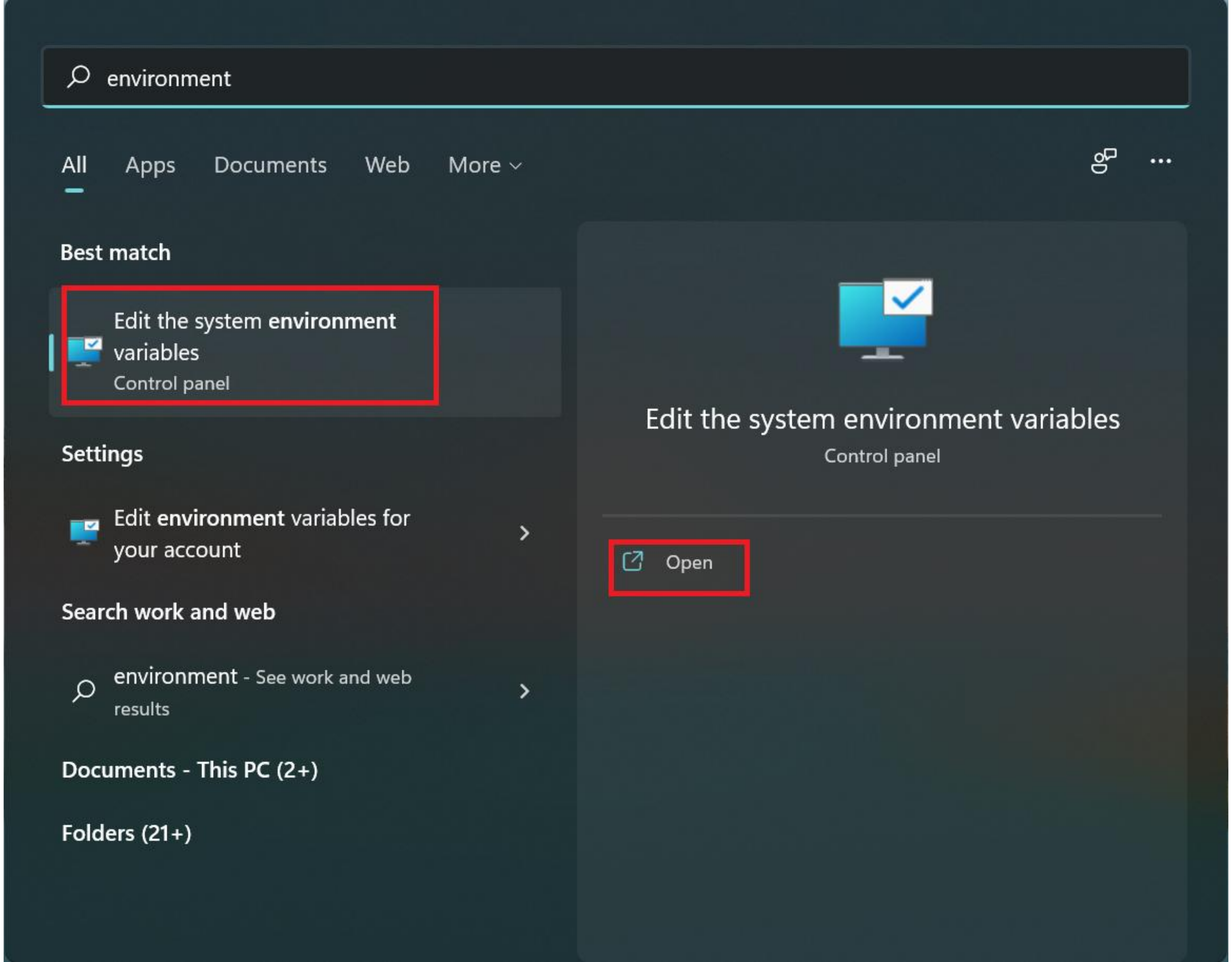


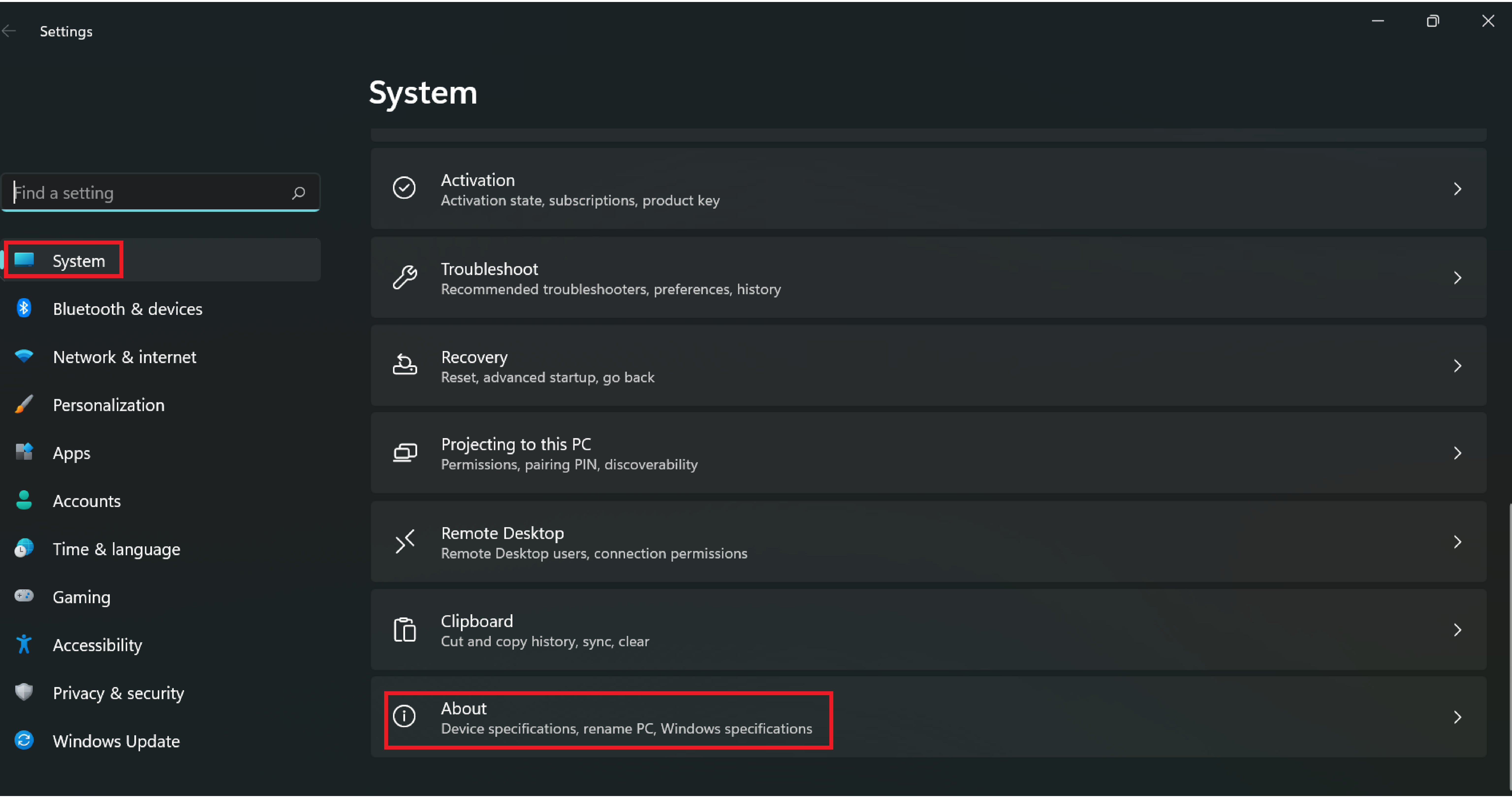
In the **Explorer** or **Command Palette (Ctrl+Shift+P)**:

- Type: `Maven: Reload Project`
- Then: `Maven: Execute Commands` → Select `clean install`

Or in the Terminal:

```bash
mvn clean install
```


environment
All
Apps
Documents
Web
More
Best match
Edit the system **environment** variables
Control panel
Settings
Edit **environment** variables for your account
Search work and web
environment - See work and web results
Documents - This PC (2+)
Folders (21+)
Edit the system environment variables
Control panel
Open



System

Activation
Activation state, subscriptions, product key

Troubleshoot
Recommended troubleshooters, preferences, history

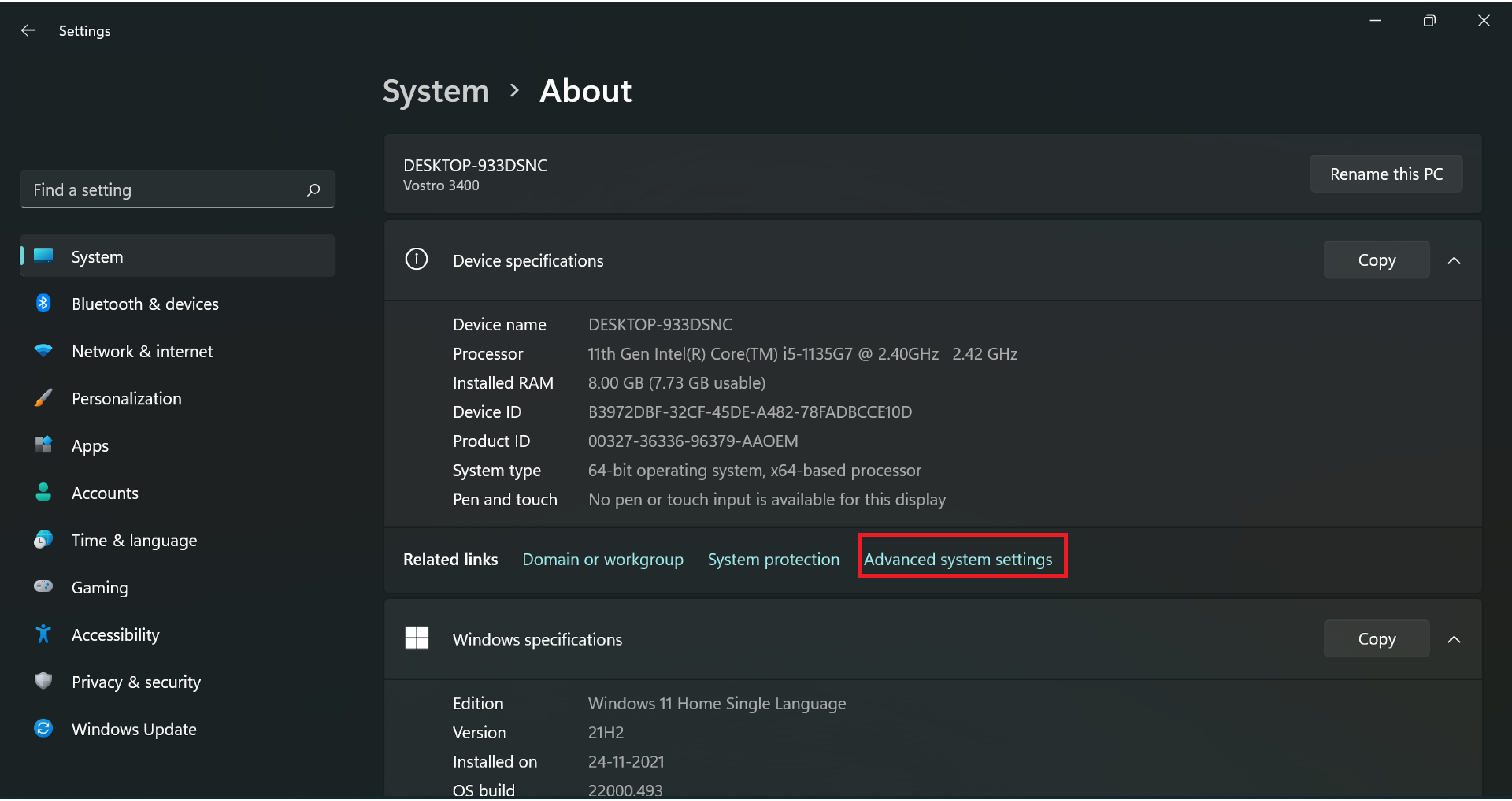
Recovery
Reset, advanced startup, go back

Projecting to this PC
Permissions, pairing PIN, discoverability

Remote Desktop
Remote Desktop users, connection permissions

Clipboard
Cut and copy history, sync, clear

About
Device specifications, rename PC, Windows specifications



System › About

DESKTOP-933DSNC
Vostro 3400

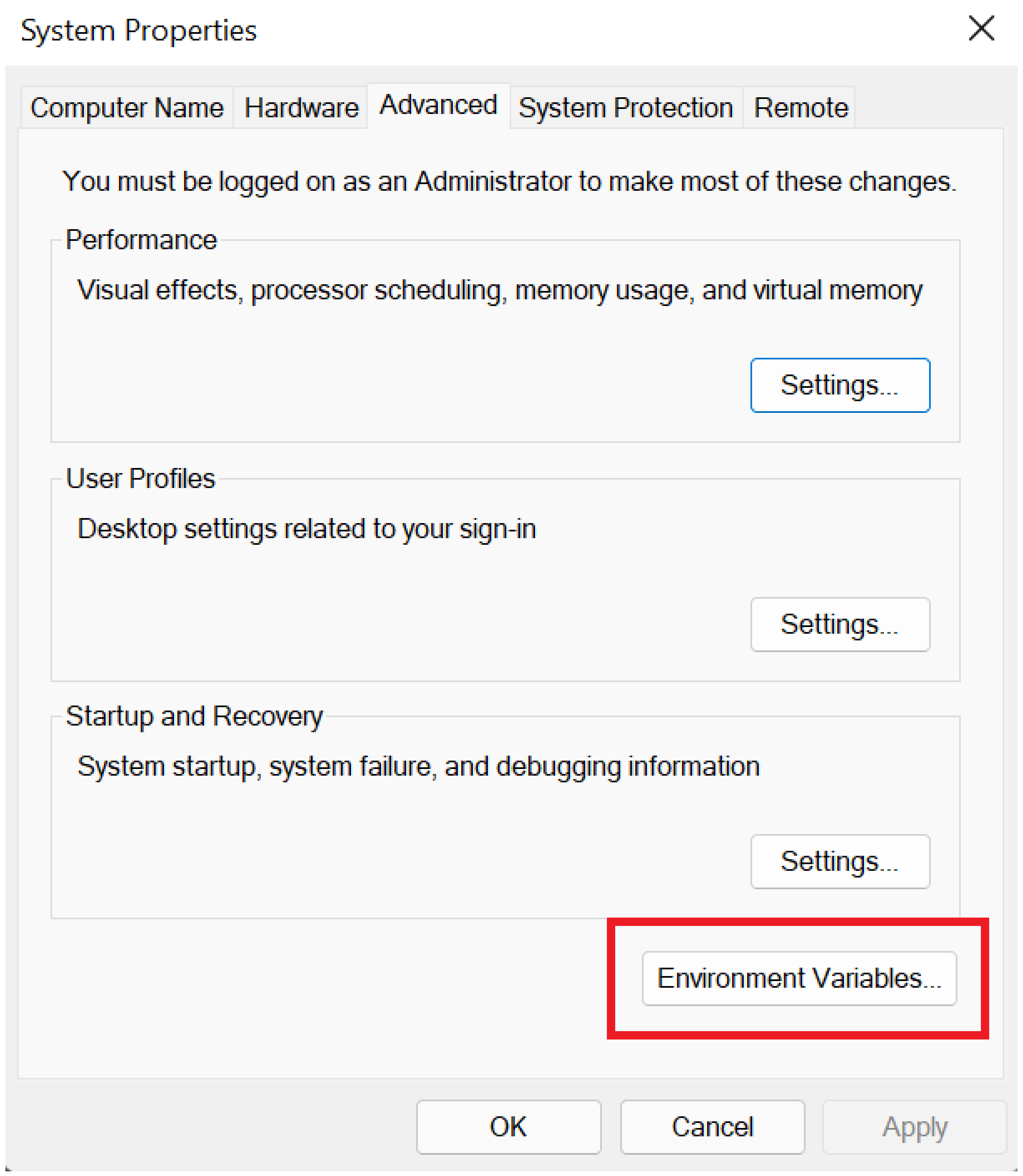
Device specifications

Device name	DESKTOP-933DSNC
Processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
Installed RAM	8.00 GB (7.73 GB usable)
Device ID	B3972DBF-32CF-45DE-A482-78FADBCCE10D
Product ID	00327-36336-96379-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Related links   Domain or workgroup   System protection   Advanced system settings

Windows specifications

Edition	Windows 11 Home Single Language
Version	21H2
Installed on	24-11-2021
OS build	22000.493


System Properties

You must be logged on as an Administrator to make most of these changes.

Performance

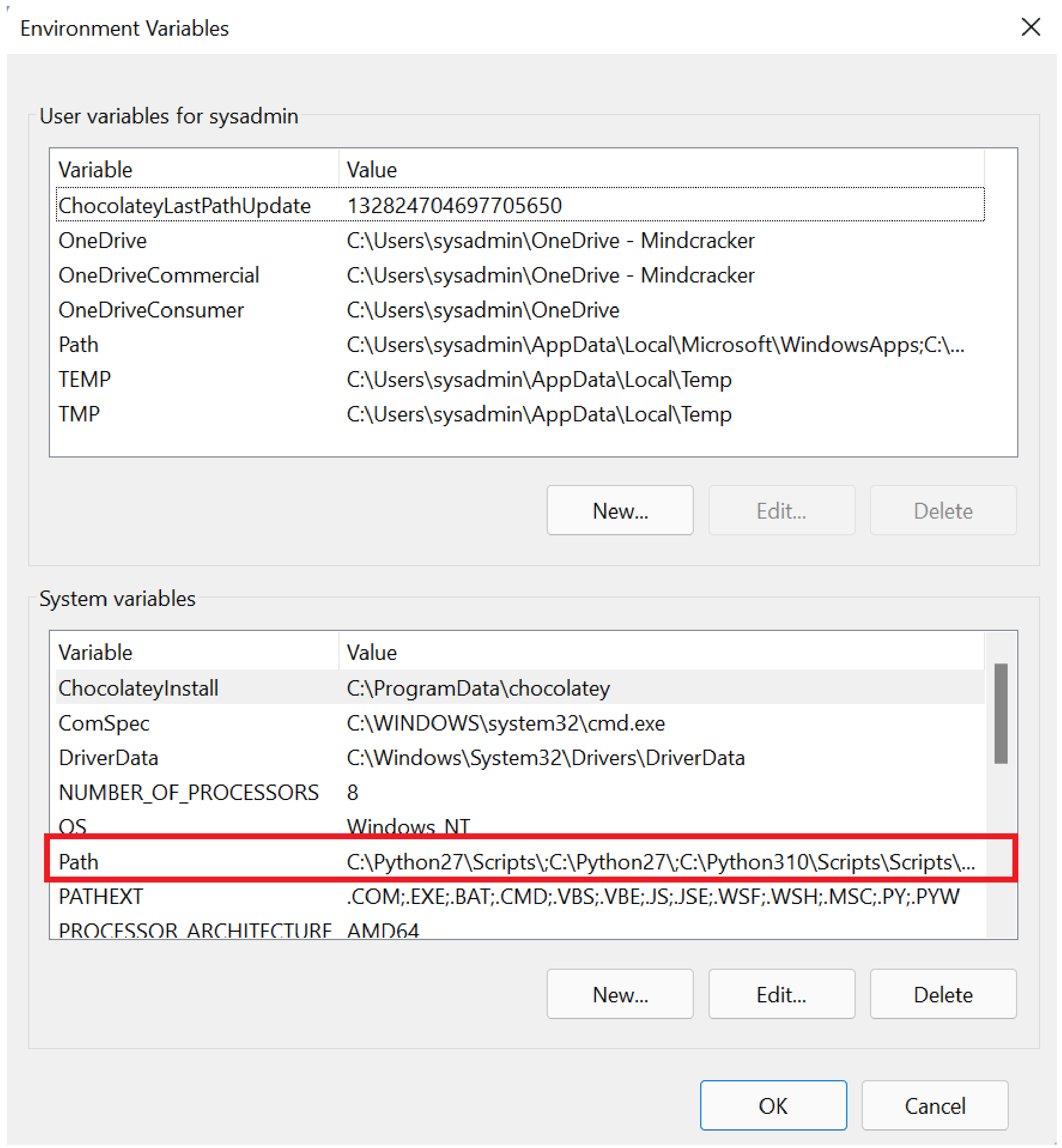
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

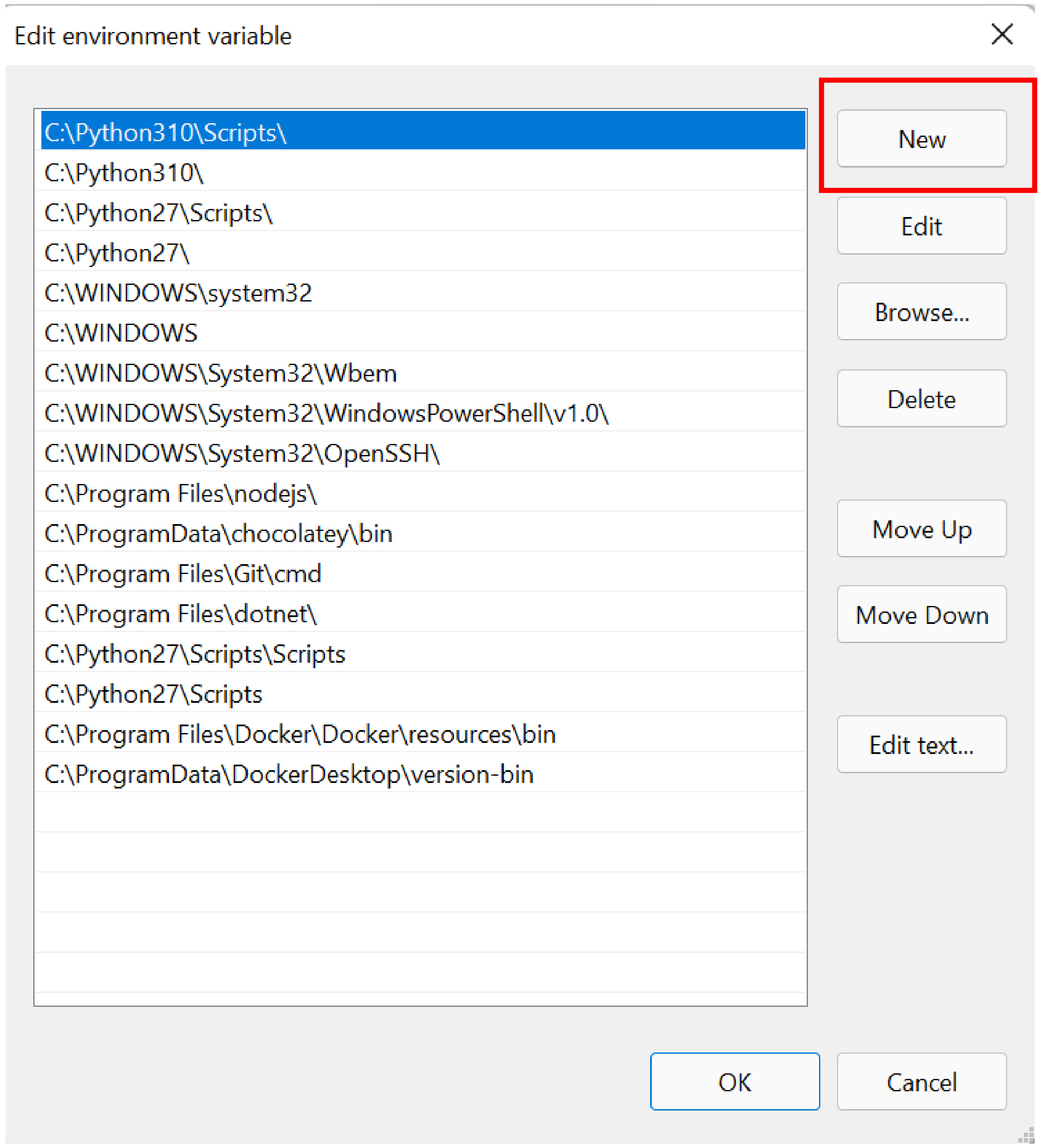

Environment Variables

User variables for sysadmin

Variable	Value
ChocolateyLastPathUpdate	132824704697705650
OneDrive	C:\Users\sysadmin\OneDrive - Mindcracker
OneDriveCommercial	C:\Users\sysadmin\OneDrive - Mindcracker
OneDriveConsumer	C:\Users\sysadmin\OneDrive
Path	C:\Users\sysadmin\AppData\Local\Microsoft\WindowsApps;C:\...
TEMP	C:\Users\sysadmin\AppData\Local\Temp
TMP	C:\Users\sysadmin\AppData\Local\Temp

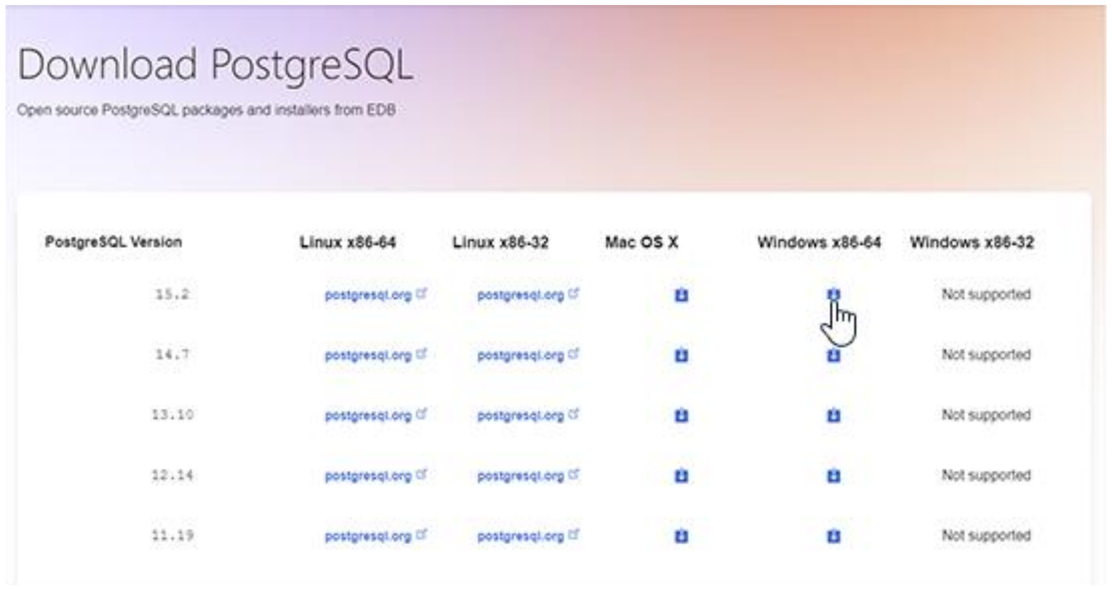
System variables

Variable	Value
ChocolateyInstall	C:\ProgramData\chocolatey
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Python27\Scripts\;C:\Python27\;C:\Python310\Scripts\Scripts\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
PROCESSOR_ARCHITECTURE	AMD64


Edit environment variable

C:\Python310\Scripts\
C:\Python310\
C:\Python27\Scripts\
C:\Python27\
C:\WINDOWS\system32
C:\WINDOWS
C:\WINDOWS\System32\Wbem
C:\WINDOWS\System32\WindowsPowerShell\v1.0\
C:\WINDOWS\System32\OpenSSH\
C:\Program Files\nodejs\
C:\ProgramData\chocolatey\bin
C:\Program Files\Git\cmd
C:\Program Files\dotnet\
C:\Python27\Scripts\Scripts
C:\Python27\Scripts
C:\Program Files\Docker\Docker\resources\bin
C:\ProgramData\DockerDesktop\version-bin

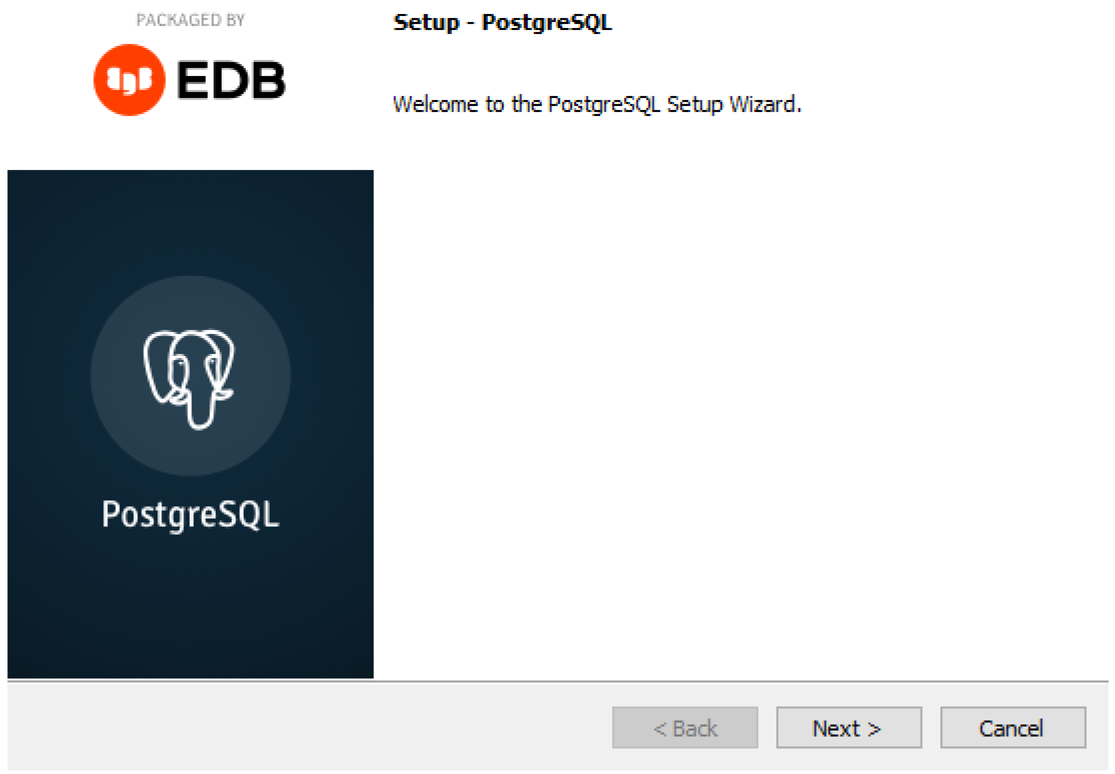
https://www.enterprisedb.com/downloads/postgres-postgresql-downloads



Download PostgreSQL

Open source PostgreSQL packages and installers from EDB

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
15.2	postgresql.org	postgresql.org	⬇	⬇	Not supported
14.7	postgresql.org	postgresql.org	⬇	⬇	Not supported
13.10	postgresql.org	postgresql.org	⬇	⬇	Not supported
12.14	postgresql.org	postgresql.org	⬇	⬇	Not supported
11.19	postgresql.org	postgresql.org	⬇	⬇	Not supported



Setup - PostgreSQL

Welcome to the PostgreSQL Setup Wizard.

https://www.w3schools.com/postgresql/postgresql_install.php

Install Java
for Visual Studio Code
2025

Project Structure Overview

Maven Project

Standard Java project structure

Source Folders

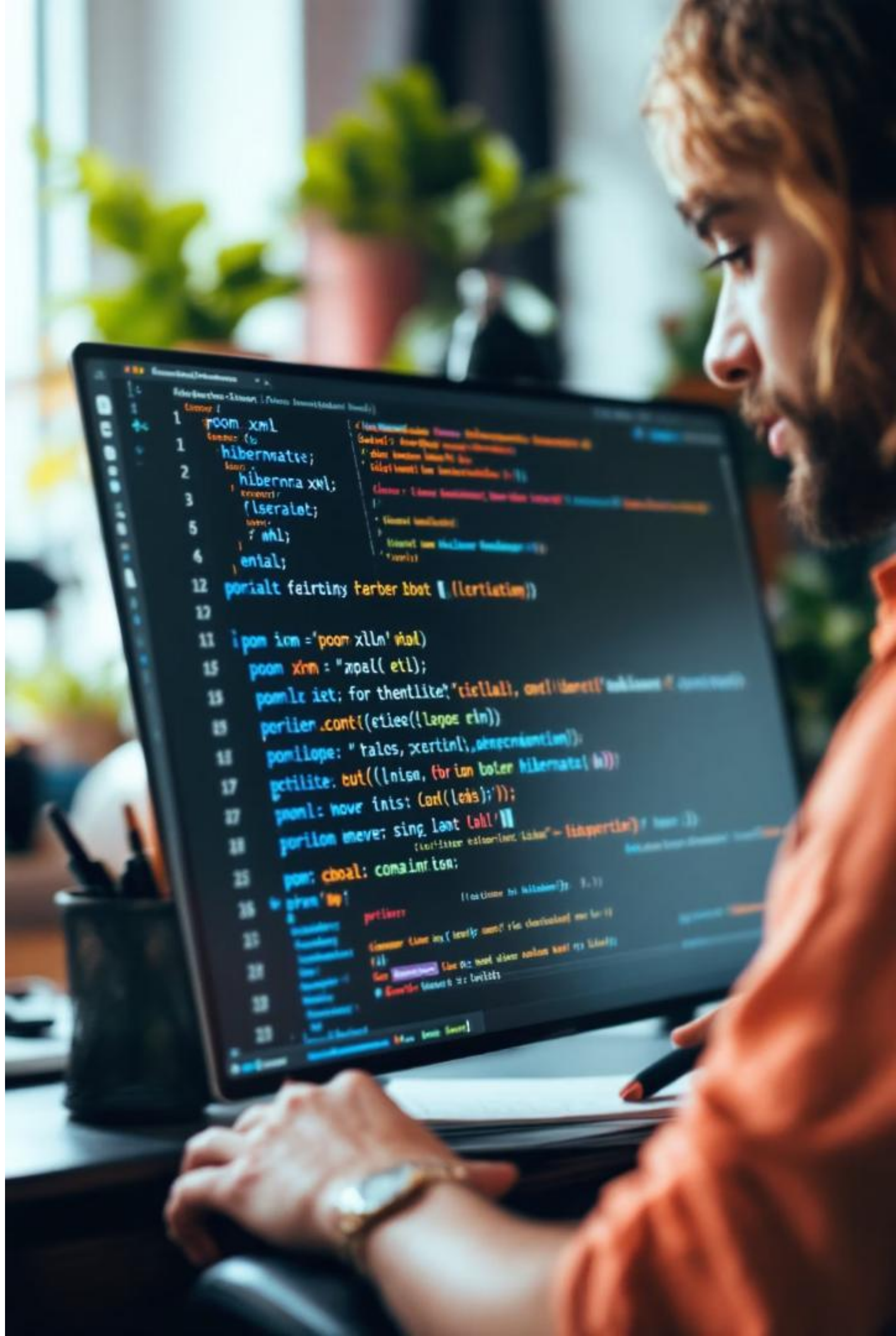
/src/main/java for code

Resources

/src/main/resources for configuration files

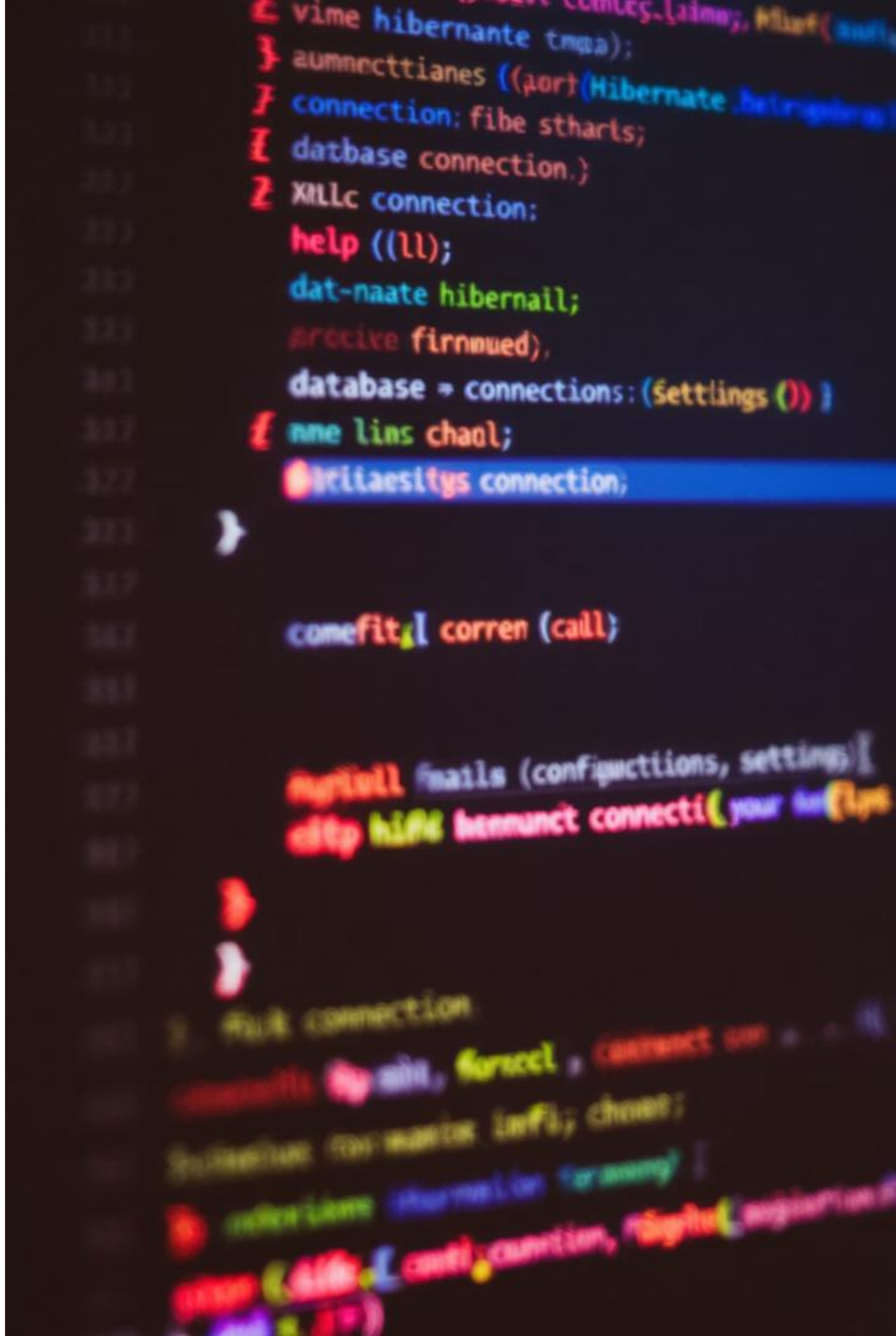
Add Dependencies (pom.xml)

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.7.4</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>7.0.0.Beta3</version>
    </dependency>

  </dependencies>
```



hibernate.cfg.xml Configuration

```
<hibernate-configuration xmlns="http://www.hibernate.org/xsd/orm/cfg">
    <session-factory>
        <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
        <property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/payroll</property>
        <property name="hibernate.connection.username">postgres</property>
        <property name="hibernate.connection.password">1234567</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="hibernate.show_sql">true</property>

    </session-factory>
</hibernate-configuration>
```



Mapping Entities

Entity Annotations

- @Entity marks Java class as entity
- @Table specifies database table

Field Annotations

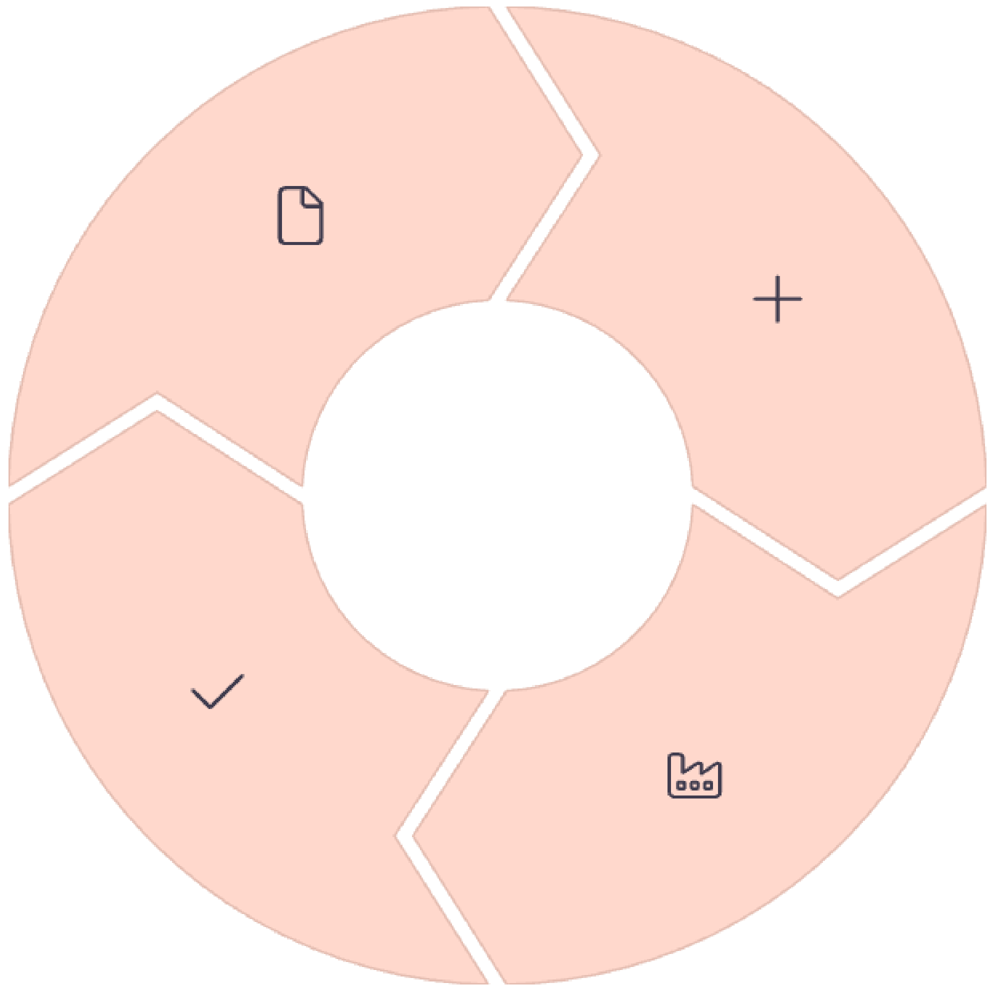
- @Id marks primary key
- @GeneratedValue for auto-increment

Relationship Annotations

- @OneToMany links related entities
- @ManyToOne defines reverse relation

Bootstrap Hibernate

```
private static final SessionFactory sessionFactory = new Configuration()
        .configure("hibernate.cfg.xml")
        .addAnnotatedClass(Employee.class)
        .addAnnotatedClass(Payroll.class)
        .buildSessionFactory();
```



Load Configuration

Read hibernate.cfg.xml

Add Classes

Register entity classes

Build Factory

Create SessionFactory

Ready

Hibernate initialized

Test Hibernate Connection

Create Test Class

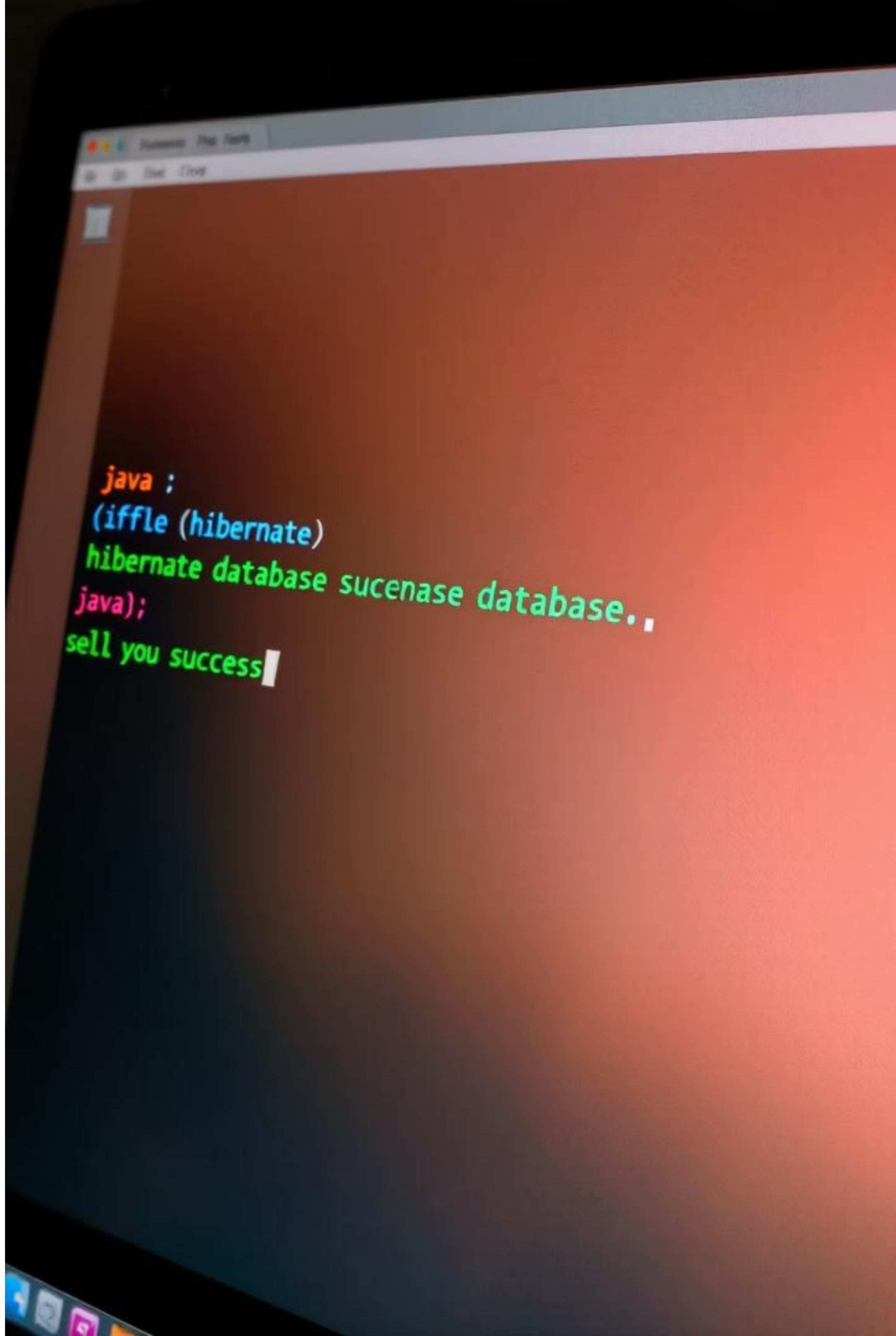
Main method to test connection

Open Session

Get session from factory

Verify

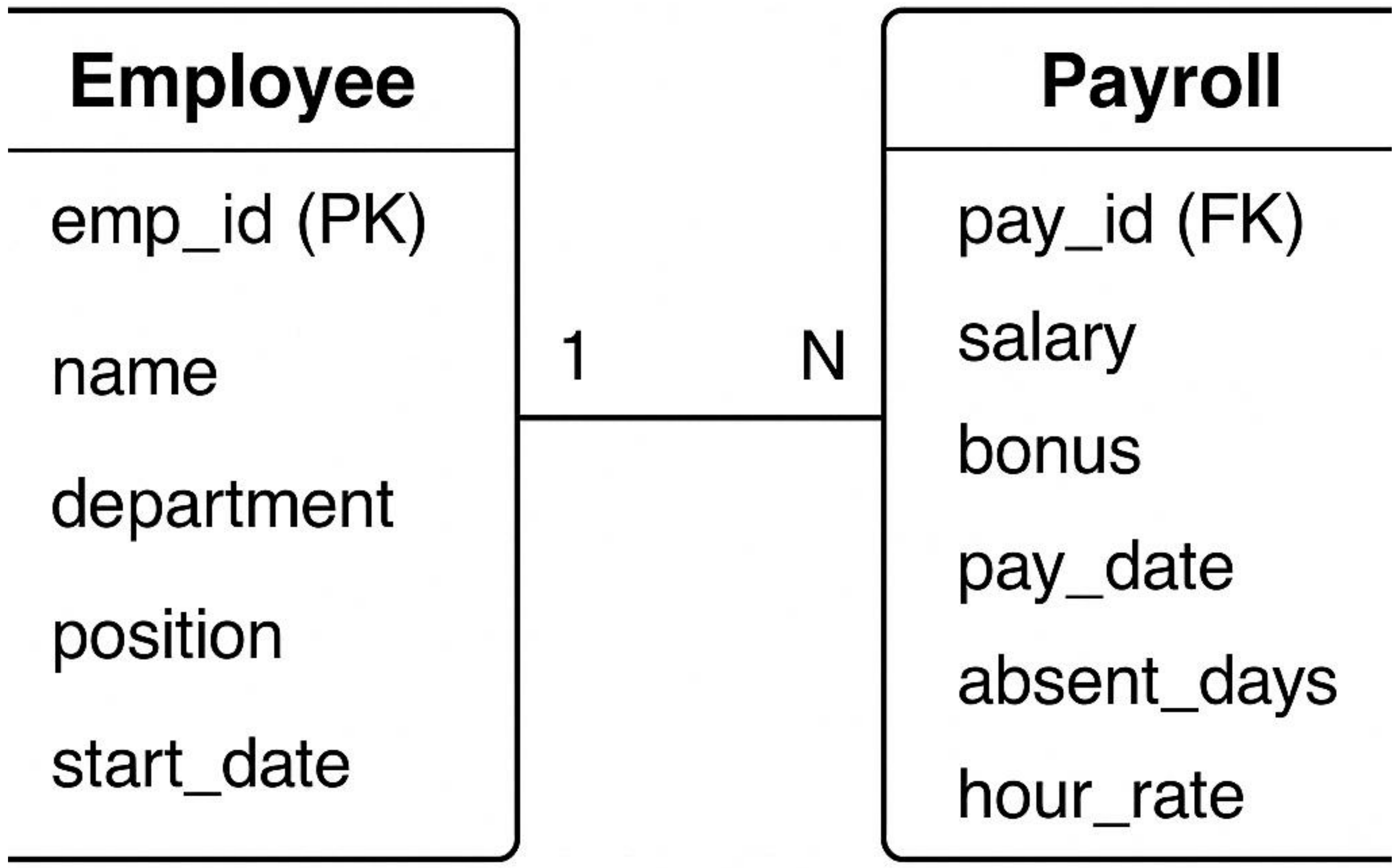
Print "Connected!" if successful



```
java ;
(iffle (hibernate)
hibernate database sucenase database.,
java);
sell you success
```

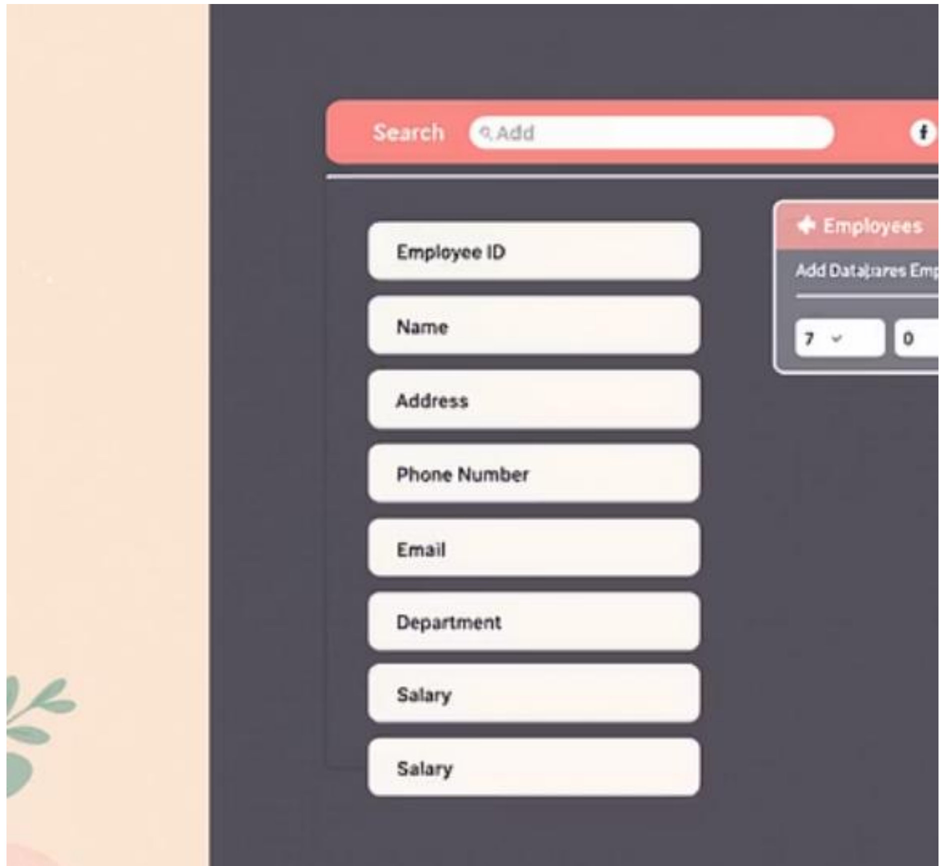
Why Create Object Classes?

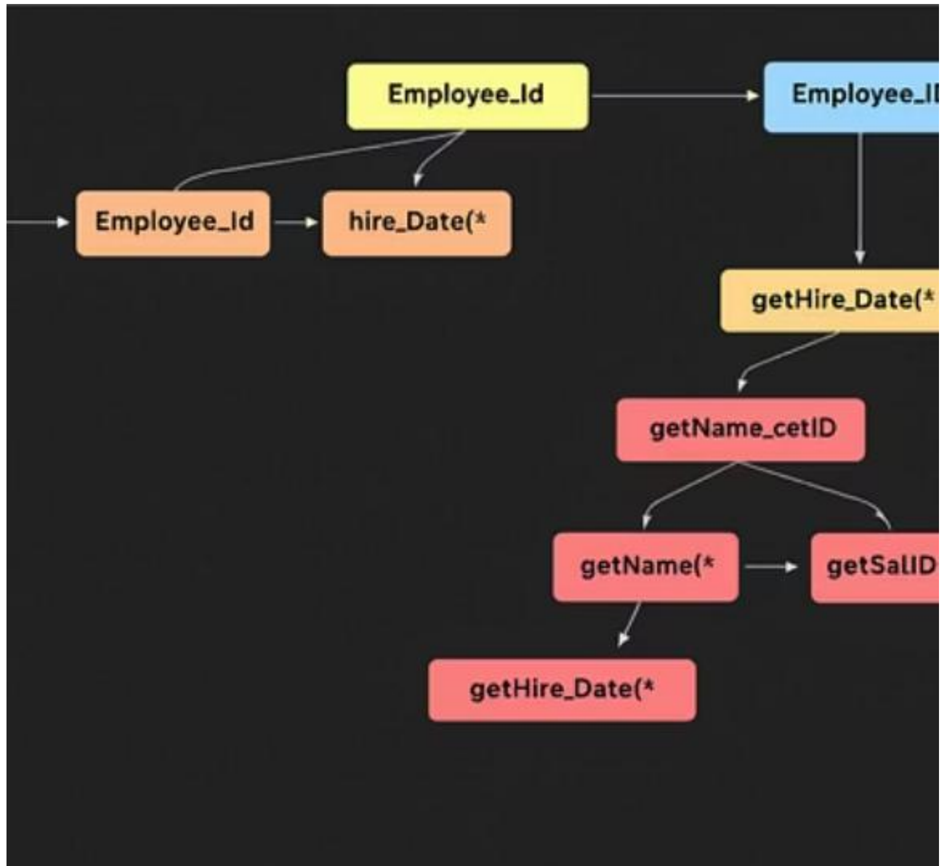
Database Structure	Maps To	Java Structure
Tables	↔	Classes
Columns	↔	Fields
Relationships	↔	Object references


Employee
emp_id (PK)
name
department
position
start_date
1
N
Payroll
pay_id (FK)
salary
bonus
pay_date
absent_days
hour_rate

Employee Class

```
@Entity
public class Employee {
  @Id
  @GeneratedValue
  private int empId;
  private String name;
  private String department;
  ...
}
```



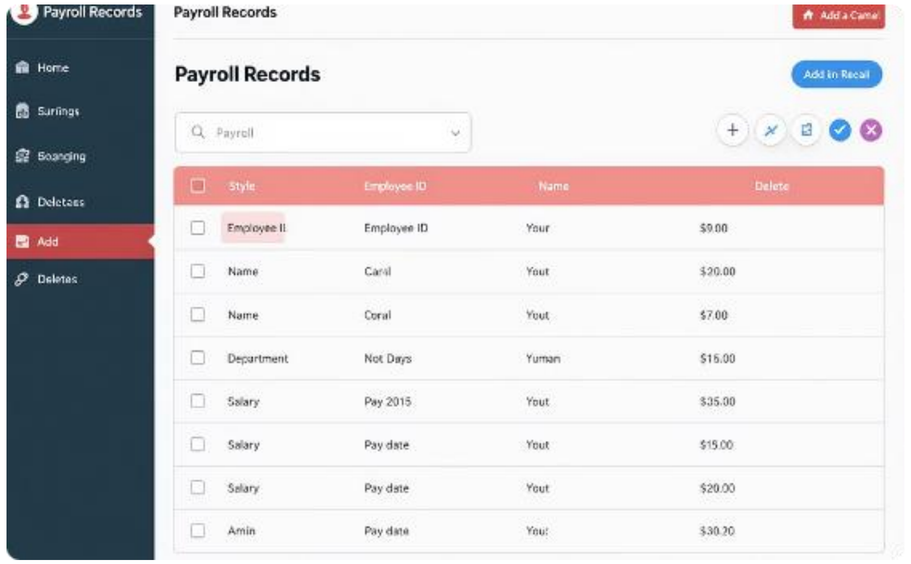


Payroll Class

```
@Entity
public class Payroll {
  @Id
  @GeneratedValue
  private int payId;
  private double salary;
  ...
}
```

Entity Class

Java class with annotations

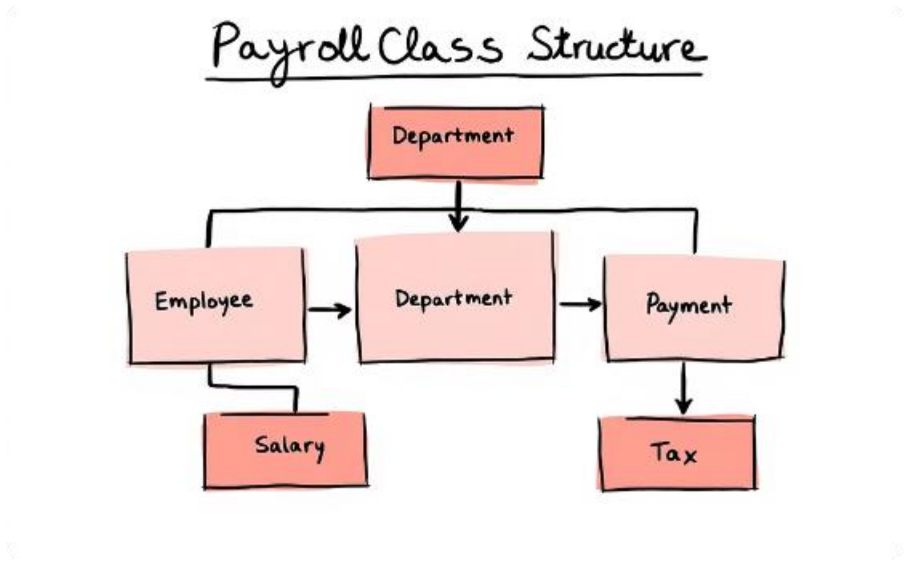


Payroll Records

Style	Employee ID	Name	Delete
Employee II	Employee ID	Your	$9.00
Name	Caril	Yout	$20.00
Name	Coral	Yout	$7.00
Department	Not Days	Yuman	$16.00
Salary	Pay 2015	Yout	$35.00
Salary	Pay date	Yout	$15.00
Salary	Pay date	Yout	$20.00
Amin	Pay date	You:	$30.20

Database Table

Corresponding SQL table



UML Design

Visual class representation

Use of Getters and Setters

JavaBean Standard

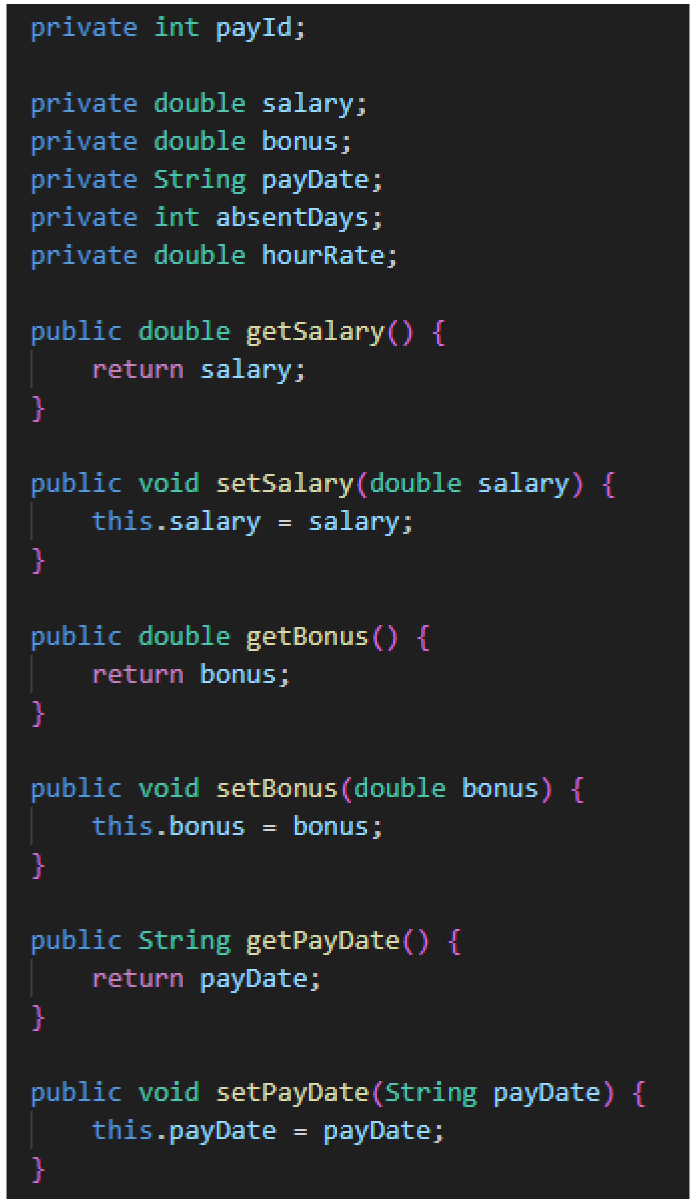
Follow established pattern for properties

Getter Methods

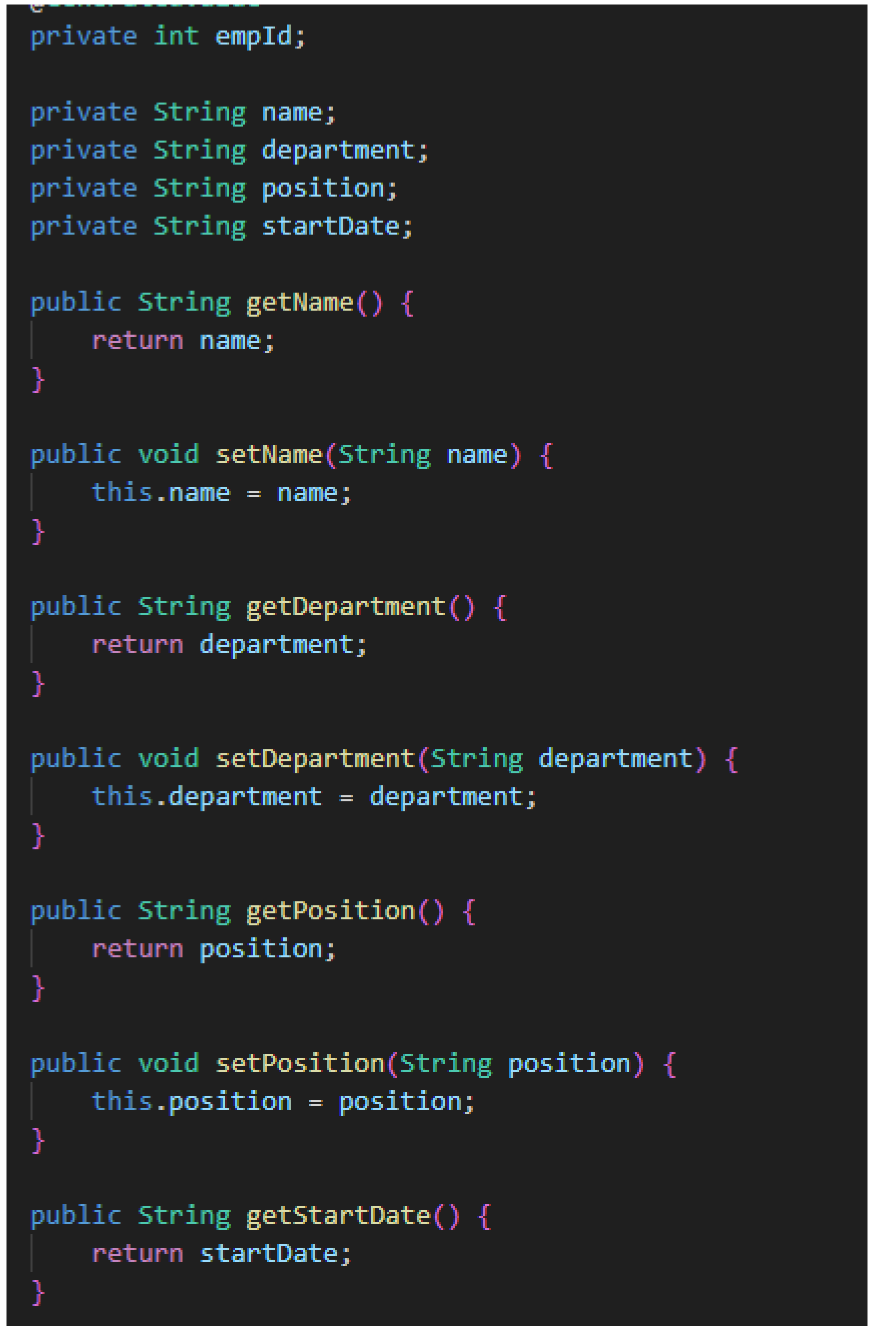
getX() returns field value

Setter Methods

setX() updates field value



```java
private int payId;

private double salary;
private double bonus;
private String payDate;
private int absentDays;
private double hourRate;

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}

public double getBonus() {
    return bonus;
}

public void setBonus(double bonus) {
    this.bonus = bonus;
}

public String getPayDate() {
    return payDate;
}

public void setPayDate(String payDate) {
    this.payDate = payDate;
}
```



```java
private int empId;

private String name;
private String department;
private String position;
private String startDate;

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getDepartment() {
    return department;
}

public void setDepartment(String department) {
    this.department = department;
}

public String getPosition() {
    return position;
}

public void setPosition(String position) {
    this.position = position;
}

public String getStartDate() {
    return startDate;
}
```

Entity Relationships

```
@ManyToOne
private Employee employee;
```

@OneToOne

One-to-one relationship

@OneToMany

One-to-many relationship

@ManyToMany

Many-to-many relationship

@ManyToOne

Many-to-one relationship

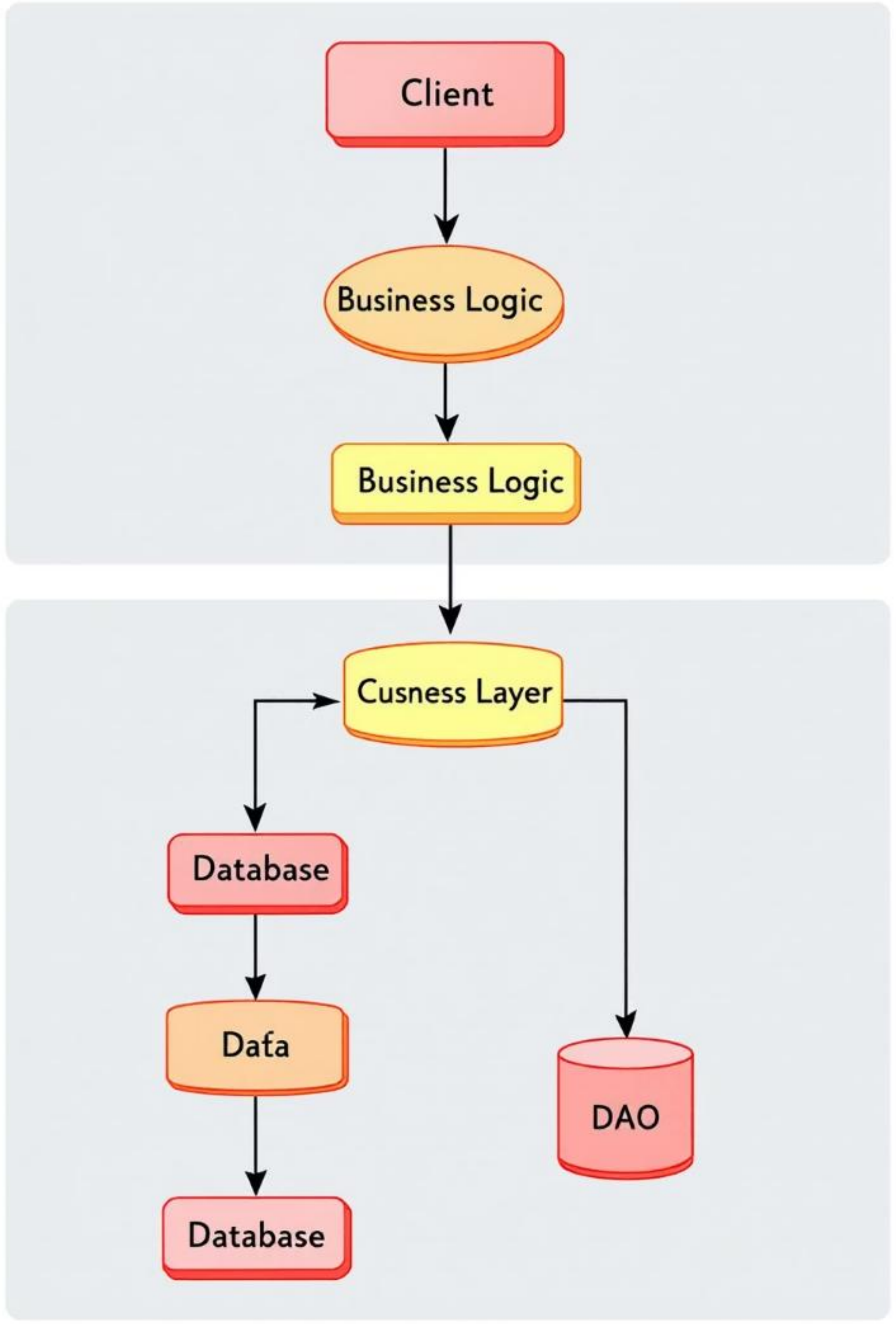
Map Employee ↔ Payroll (1:N)

Employee Side

```
@OneToMany(mappedBy="employee")
private List payrollList;
```

Payroll Side

```
@ManyToOne
private Employee employee;
```



DAO Pattern

Data Access Object

Encapsulates database operations

Separation of Concerns

Isolates data access logic

Reusable Code

Centralizes database operations

Add Employee (DAO)

Create Object

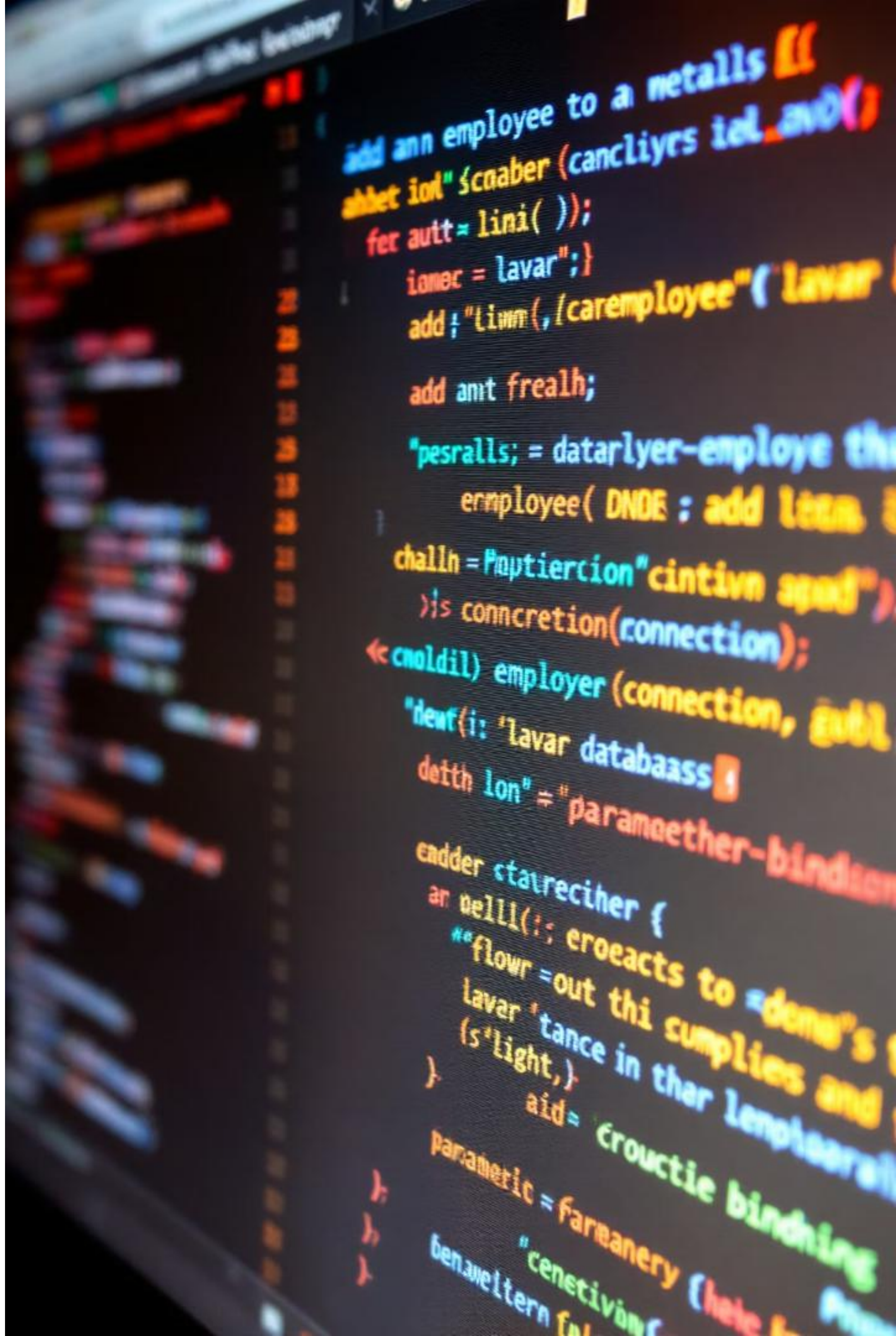
Instantiate Employee with data

Open Session

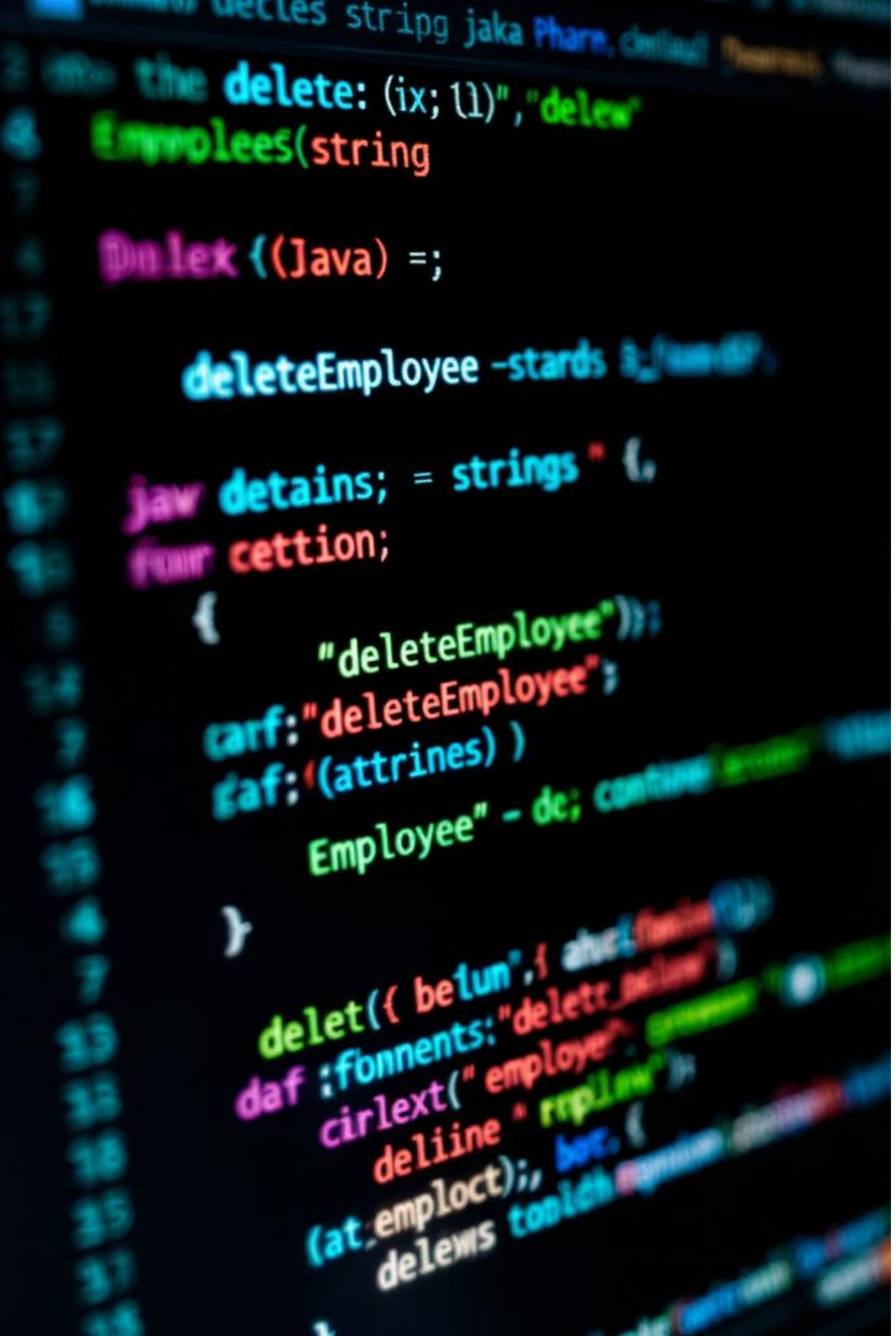
Get Hibernate session

Save Object

session.save(employee);



```java
public void addEmployee(String name, String department, String position, String startDate) {
        Transaction tx = null;

        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            tx = session.beginTransaction();

            // Step 1: Create Object
            Employee employee = new Employee();
            employee.setName(name);
            employee.setDepartment(department);
            employee.setPosition(position);
            employee.setStartDate(startDate);

            // Step 2: Save Object
            session.persist(employee);  // Use persist() or save()

            tx.commit();
            System.out.println("✅ Employee added: " + employee);
        } catch (Exception e) {
            if (tx != null) tx.rollback();
            e.printStackTrace();
        }
    }
```



Delete Employee (DAO)

Find Employee

Locate by ID

Delete Object

session.delete(employee);

Commit Transaction

Save changes to database

```
public void deleteEmployeeById(int empId) {
        Transaction tx = null;

        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            tx = session.beginTransaction();

            // Step 1: Find
            Employee emp = session.find(Employee.class, empId);

            if (emp != null) {
                // Step 2: Delete
                session.remove(emp);  // ✅ Hibernate 6+

                System.out.println("✅ Employee deleted: " + empId);
            } else {
                System.out.println("⚠️ Employee not found: " + empId);
            }

            // Step 3: Commit
            tx.commit();
        } catch (Exception e) {
            if (tx != null) tx.rollback();
            e.printStackTrace();
        }
    }
```

List All Employees

```
List list = session.createQuery("from Employee").list();
```

1

Create Query

HQL query to select all employees

2

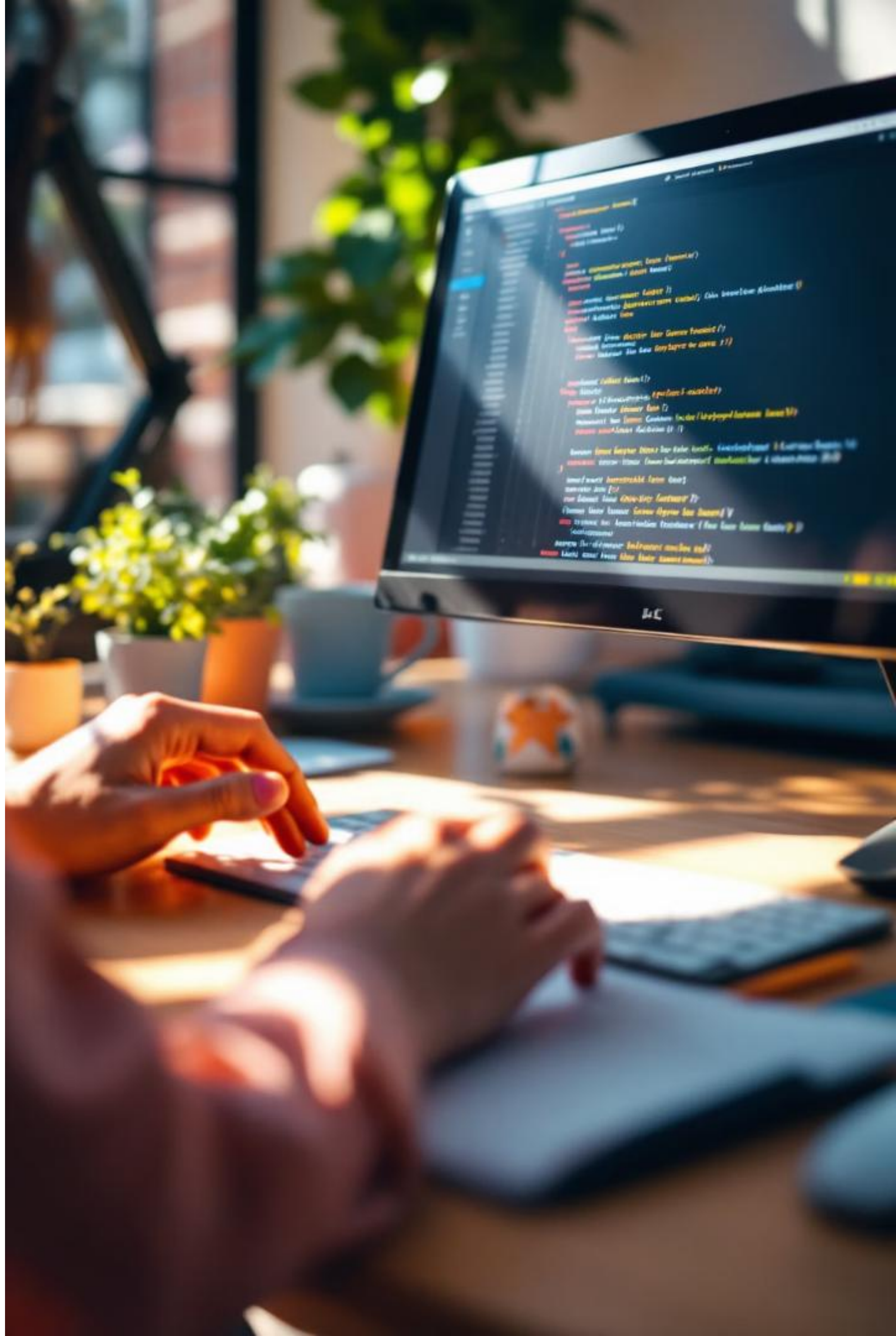
Execute Query

Get result list

3

Process Results

Iterate through employee objects



```
public void listAllEmployees() {
    try (Session session = HibernateUtil.getSessionFactory().openSession()) {

      // Step 1: Create Query (HQL)
      String hql = "FROM Employee"; // HQL uses class name, not table name
      List<Employee> employees = session.createQuery(hql, Employee.class).list();

      // Step 2: Process Results
      if (employees.isEmpty()) {
          System.out.println("⚠️ No employees found.");
      } else {
        for (Employee emp : employees) {
            System.out.println("🧑‍💼 " + emp);
        }
      }

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
```

Add Payroll

```
Payroll p = new Payroll(...);
session.save(p);
```

±

Create Payroll

New Payroll object with data

2

Open Session

Get Hibernate session

Save Record

Persist to database

```java
public void addPayroll(double salary, double bonus, String payDate, int absentDays, double hourRate, Employee employee) {
        Transaction tx = null;

        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            tx = session.beginTransaction();

            // Step 1: Create Payroll Object
            Payroll payroll = new Payroll();
            payroll.setSalary(salary);
            payroll.setBonus(bonus);
            payroll.setPayDate(payDate);
            payroll.setAbsentDays(absentDays);
            payroll.setHourRate(hourRate);

            // Associate with employee
            payroll.setEmployee(employee);

            // Step 2: Persist to DB
            session.persist(payroll);

            tx.commit();
            System.out.println("✅ Payroll added: " + payroll);
        } catch (Exception e) {
            if (tx != null) tx.rollback();
            e.printStackTrace();
        }
    }
```

Set Payroll to Employee

```
payroll.setEmployee(employee);
employee.getPayrollList().add(payroll);
```

Link Objects

Connect both sides of relationship

Save Changes

Persist updated objects

Verify

Confirm relationship established

```
public static void main(String[] args) {
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            Transaction tx = session.beginTransaction();

            // Re-fetch employee in the same session
            Employee emp = session.find(Employee.class, 1);

            if (emp != null) {
                Payroll payroll = new Payroll();
                payroll.setSalary(25000);
                payroll.setBonus(3000);
                payroll.setPayDate("2025-04-30");
                payroll.setAbsentDays(2);
                payroll.setHourRate(150.0);

                emp.addPayroll(payroll);  // No LazyInitializationException

                session.merge(emp);
                tx.commit();
                System.out.println("✅ Payroll added for employee ID: " + emp.getEmpId());
            }
        }

    }
```

```
SELECT table_schema, table_name
FROM information_schema.tables
WHERE table_type = 'BASE TABLE'
  AND table_schema NOT IN ('pg_catalog', 'information_schema');

select * from employee;

select * from payroll;
```

Cascade and Fetch Types

Cascade Types

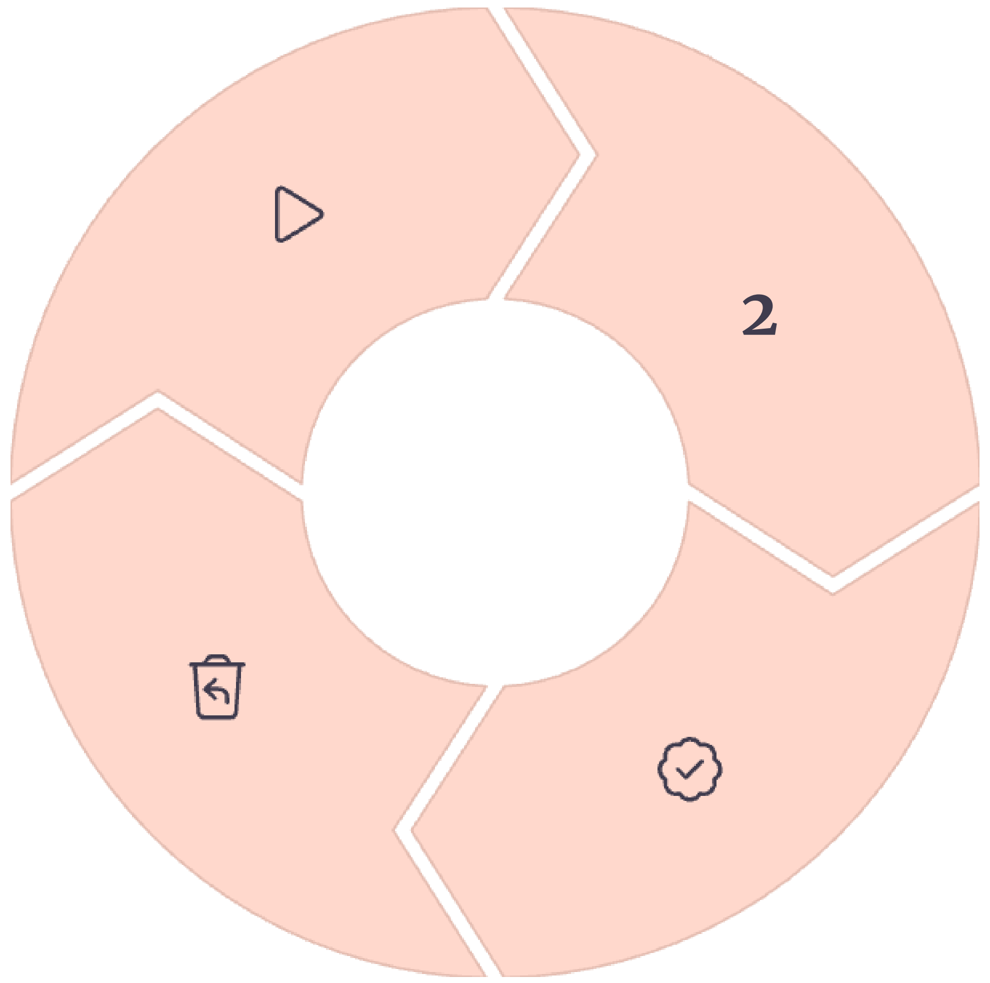
- CascadeType.ALL
- CascadeType.PERSIST
- CascadeType.REMOVE

Fetch Types

- FetchType.LAZY
- FetchType.EAGER

Transaction Management

```
Transaction tx = session.beginTransaction();
// operations
tx.commit();
```



Begin

Start transaction

Operate

Perform database operations

Rollback

Revert on error

Commit

Save changes permanently

Error Handling

```
try {
  // Hibernate operations
} catch (Exception e) {
  e.printStackTrace();
}
```

Try Block

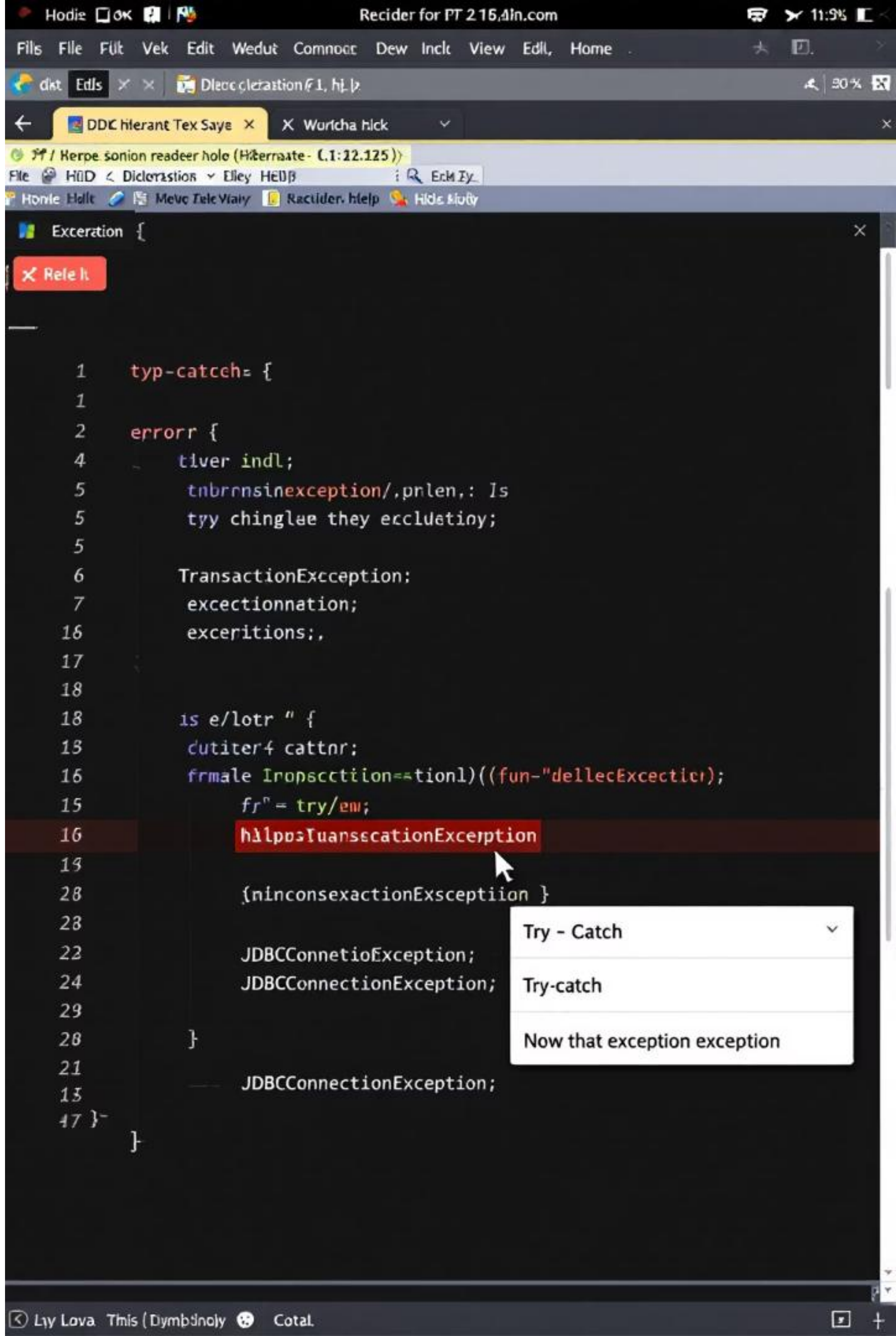
Attempt database operations

Catch Exceptions

Handle specific error types

Rollback Transaction

Revert changes on error


Hodie OK
Recider for PT 2 15.4ln.com
11:9%
Fils File Fűt Vek Edit Wedut Comnoc Dew Inck View Edl, Home
dkt Edls
Dlecc cletastion f 1, hl, b
30%
DDC hlerant Tex Saye
Worlcha hick
Kerpe sonion readeer hole (Hibernate- (.1:22.125))
File HDD Diclorastion Elley HEDB Eck Ey
Honle Hallt Meve Tele Vialy Racilder. hlelp Hids Multy
Exception {
Rele lt
Try - Catch
Try-catch
Now that exception exception
Lly Lova This (Dymbinoly Cotal

```
typ-catcch= {

errorr {
    tiver indl;
     tnbrrnsinexception/,pnlen,: Is
     tpy chinglue they eccluetiny;

    TransactionExcception:
     excectionnation;
     exceritions;,


    is e/lotr " {
     cutiterf cattnr;
     frmale Iropscction==tionl)((fun-"dellecExcectic);
        fr" = try/ew;
        hlipostuansscationException

        {ninconsexactionExsceptiion }

        JDBCConnetioException;
        JDBCConnectionException;

    }

        JDBCConnectionException;
}-
}
```

Method: print_List()

```
public void printList() {
  for (Employee e : employees)
    System.out.println(e);
}
```

Iterate Collection

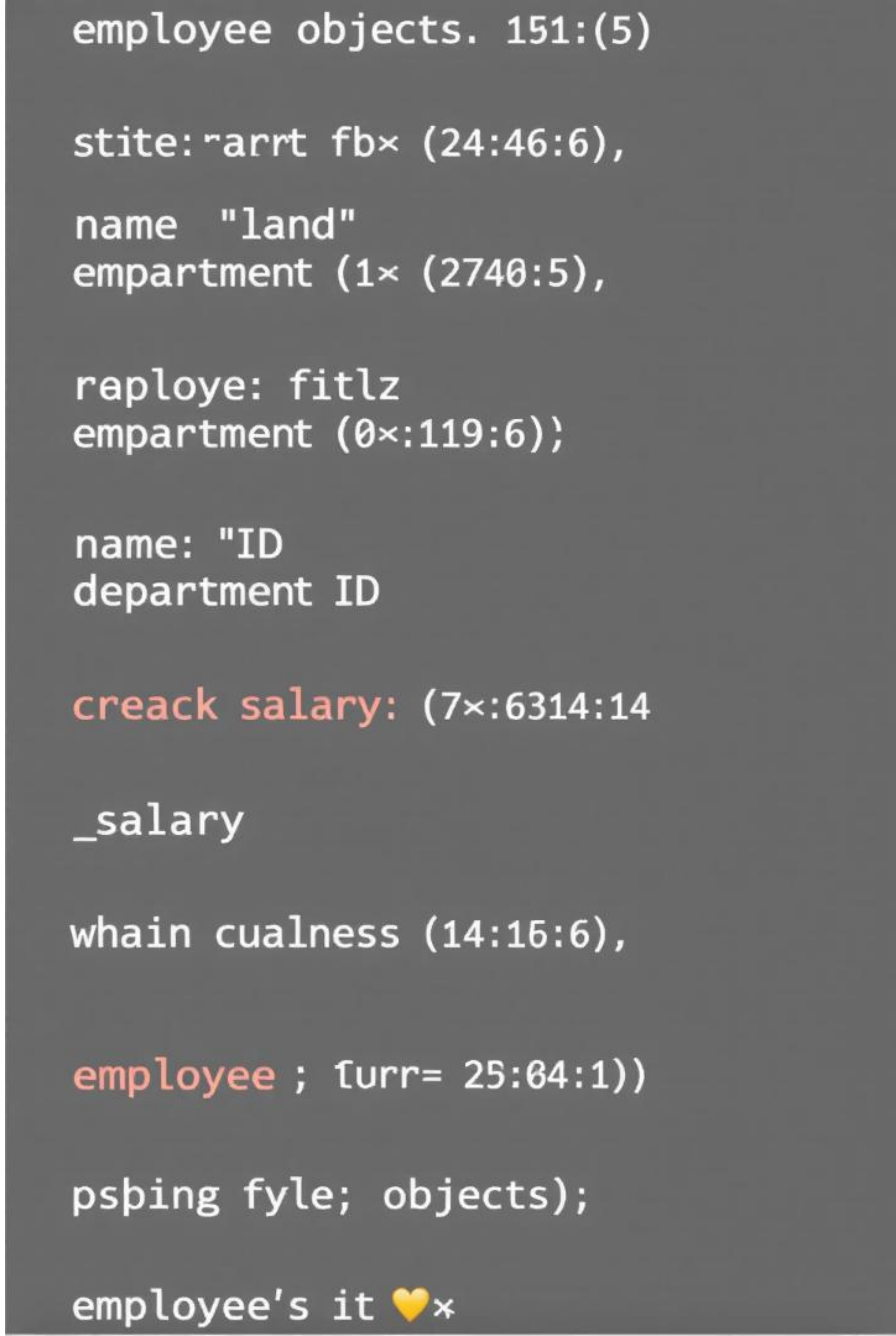
Loop through employee list

Print Each Item

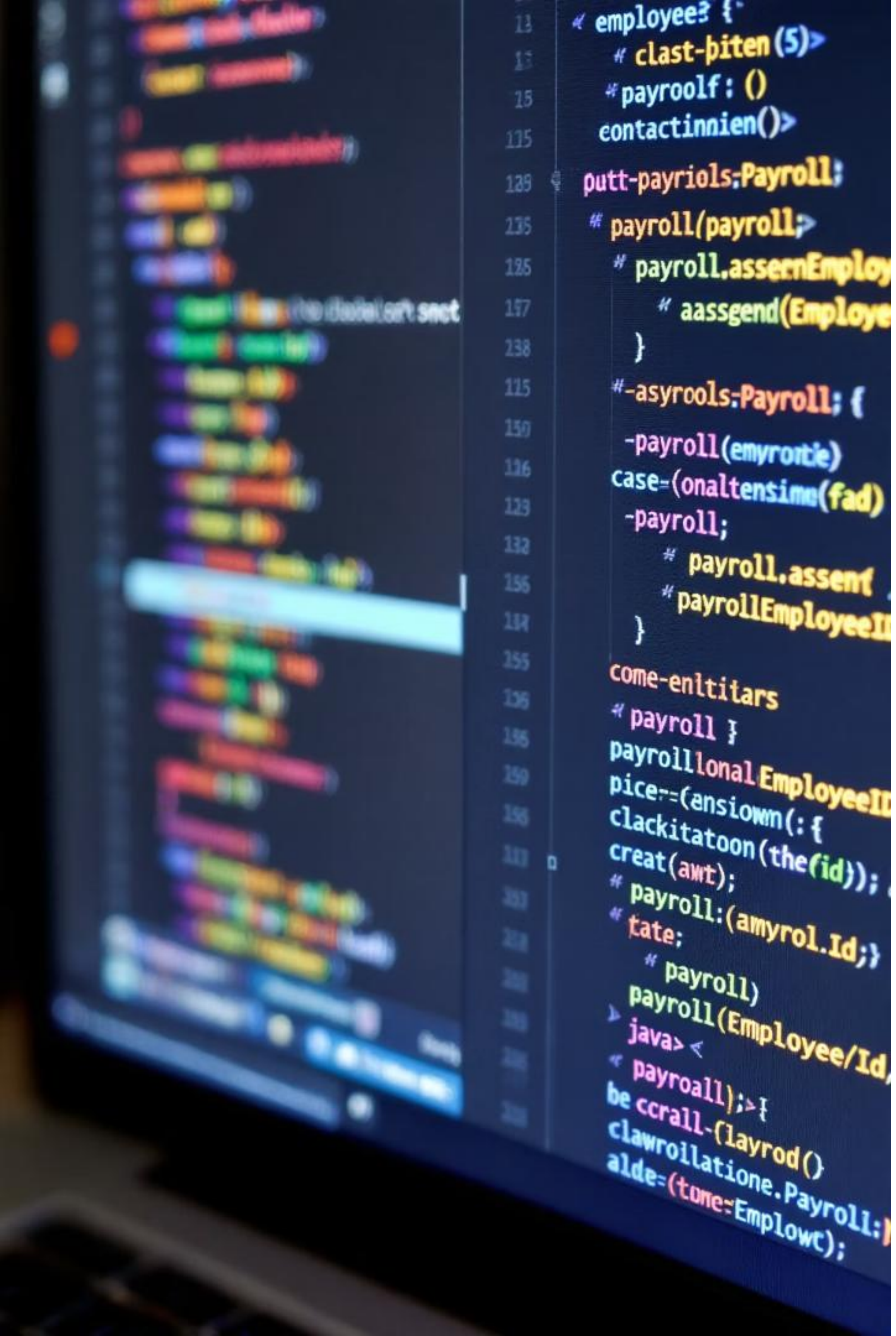
Display employee details

Console Output

View results in terminal



```
employee objects. 151:(5)

stite: ⌐arrt fb× (24:46:6),

name  "land"
empartment (1× (2740:5),

raploye: fitlz
empartment (0×:119:6);

name: "ID
department ID

creack salary: (7×:6314:14

_salary

whain cualness (14:16:6),

employee ; ſurr= 25:64:1))

psþing fyle; objects);

employee's it 💛×
```



Method: set_Payroll()

```
public void setPayroll(Employee e, Payroll p) {
  p.setEmployee(e);
}
```

Employee Parameter

Target employee object

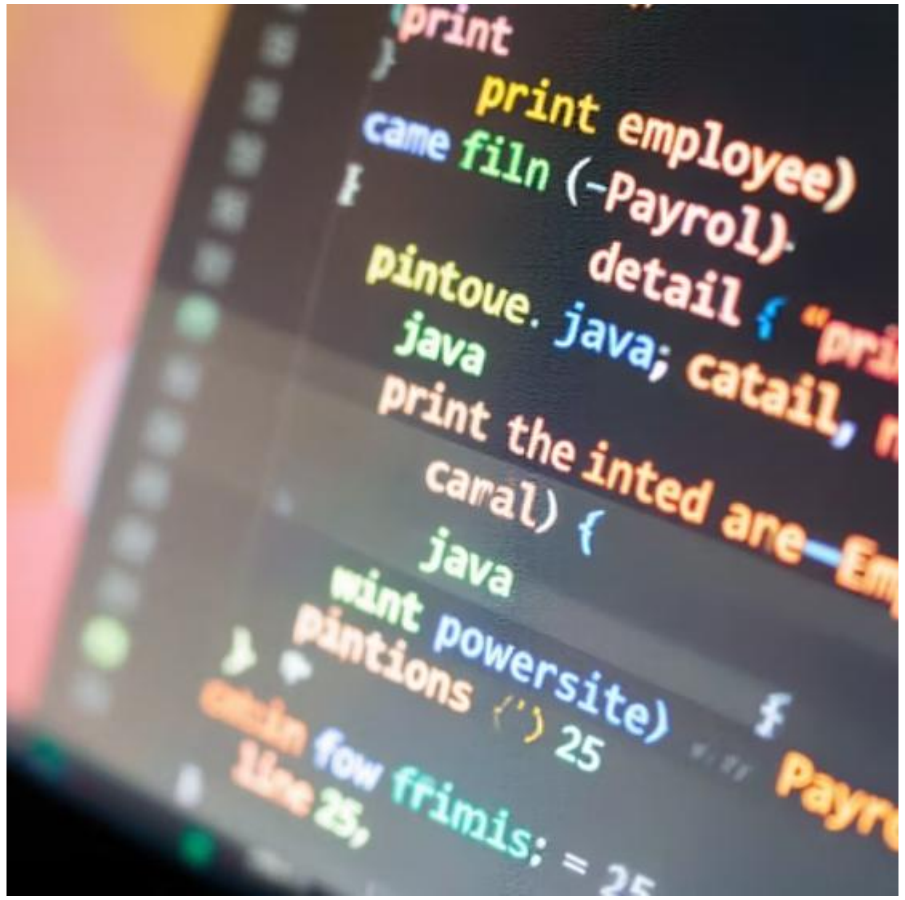
Payroll Parameter

Payroll record to associate

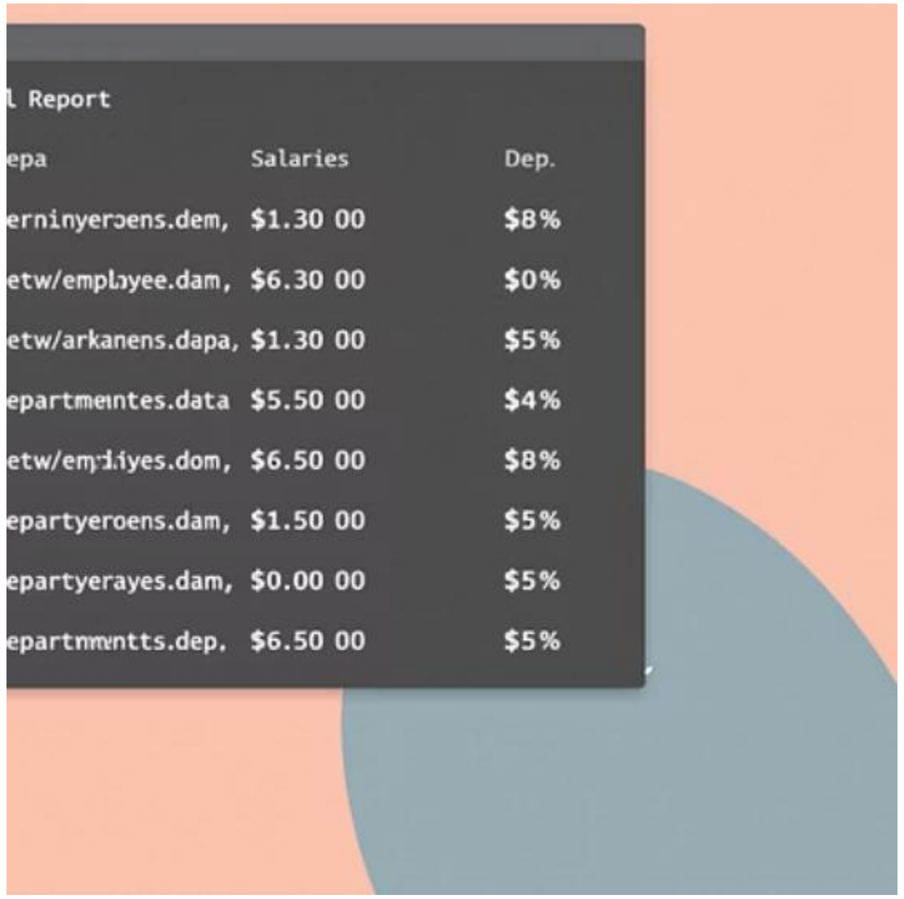
Association

Connect objects via reference

Method: print_Result()



```
print
    print employee)
came filn (-Payroll)
          detail { "pri
pintoue. java; catail,
 java
print the inted are—Em
   caral) {
    java
wint powersite)    {
pintions (') 25      Payr
cmin fow frimis; = 25
line 25,
```



```
l Report

epa                     Salaries          Dep.

erninyerɔens.dem,  $1.30 00          $8%
etw/emplayee.dam,  $6.30 00          $0%
etw/arkanens.dapa, $1.30 00          $5%
epartmeıntes.data  $5.50 00          $4%
etw/emɉiiyes.dom,  $6.50 00          $8%
epartyeroens.dam,  $1.50 00          $5%
epartyerayes.dam,  $0.00 00          $5%
epartmmntts.dep.   $6.50 00          $5%
```

```
oulin Datbass query:
Employee table    fin employee ID      Hins a camerri
                  Employee name  =     Pay date:
Joins :           Department:          Departven:
Salary: 17        Department:          pay date
Salary 19+)       Salaryier (ha(0):    5:0))
Salary 19+)
```

Combine Objects

Show employee with payroll

Format Output

Present data clearly

Display Results

Print to console

Method: get_Absent()

```
return this.absentDays;
```

1

Method Purpose

Retrieve absence data

2

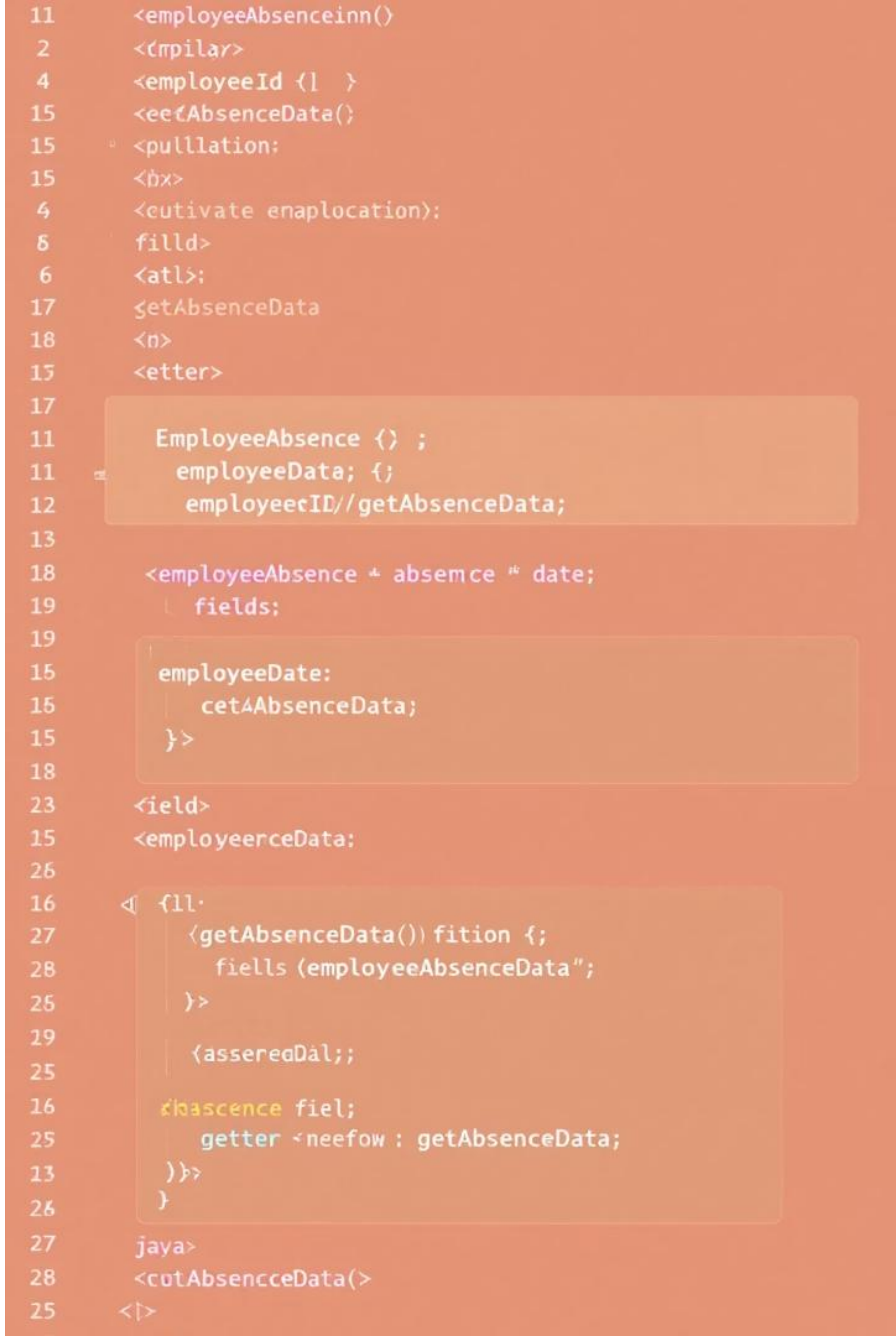
Implementation

Simple getter method

3

Usage

Calculate attendance metrics

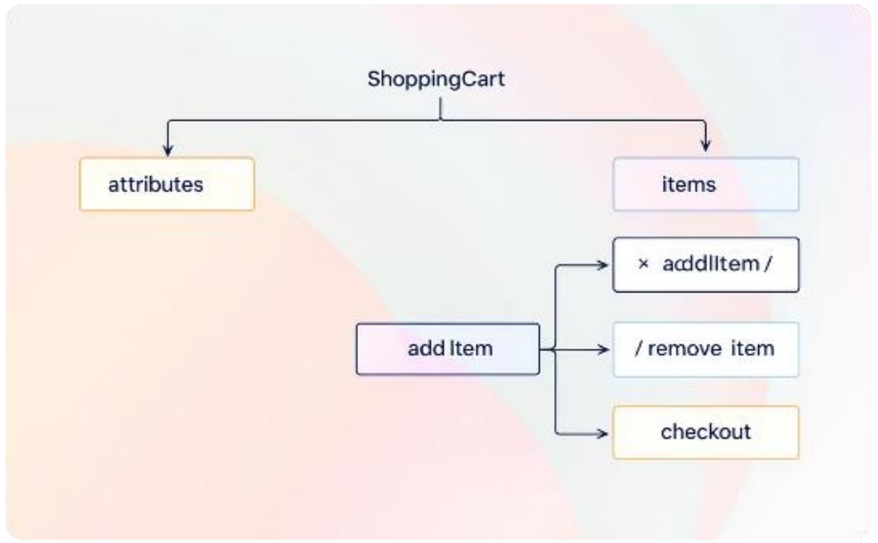


```
<employeeAbsenceinn()
<(mpilar>
<employeeId {l  }
<ee{AbsenceData()
<pullation:
<bx>
<cutivate enaplocation):
filld>
<atl>:
getAbsenceData
<n>
<etter>

  EmployeeAbsence {} ;
   employeeData; {;
    employeeID//getAbsenceData;

 <employeeAbsence + absemce * date;
    fields;

  employeeDate:
     cet4AbsenceData;
   }>

<ield>
<employeerceData:

 {ll·
   (getAbsenceData()) fition {;
     fiells (employeeAbsenceData";
   }>

   (assereaDal;;

  chascence fiel;
     getter <neefow : getAbsenceData;
   ))>
   }
jaya>
<cutAbsencceData(>
<|>
```

Method: set_hour_rate()

```
this.hourRate = rate;
```

```
{

Employee
{
  employee wills emopley lake,
  private Double,
  hourly·ate:"
  field ⅄ hourly Rate:}
}

{  setHourlyRate
  /setHourlyRate". // Sets the hourly rate > off of the employee;}
}
```



Method Code

Simple field assignment

Business Context

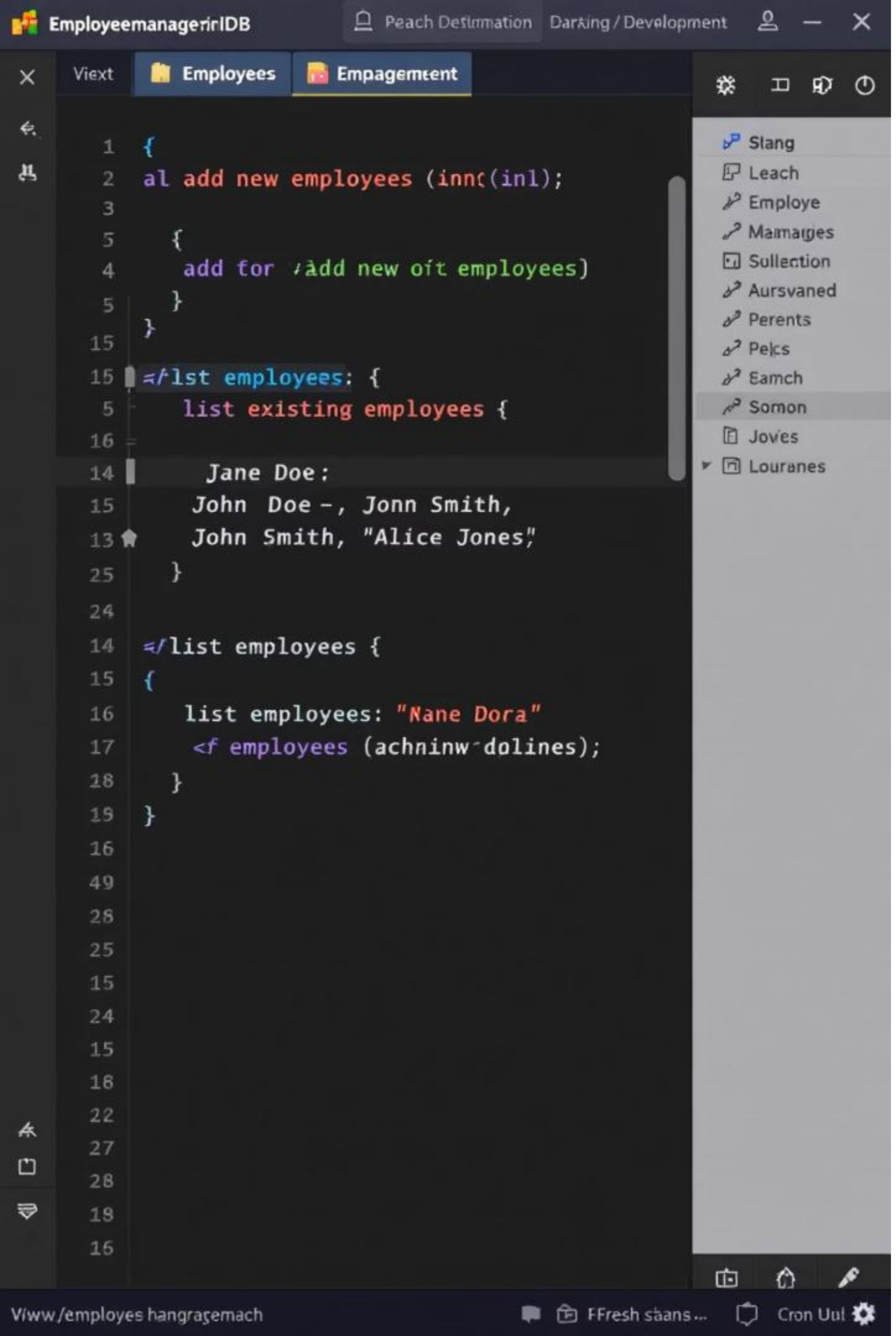
Used in payroll calculations

Design View

Part of Payroll class

Method: get_hour_rate()

```
return this.salary / totalHours;
```



```
{
al add new employees (innc(inl);

  {
   add for ⁄add new oft employees)
  }
}

≈/lst employees: {
   list existing employees {

     Jane Doe:
    John  Doe -, Jonn Smith,
    John Smith, "Alice Jones";
  }

≈/list employees {
{
   list employees: "Nane Dora"
    <f employees (achninw-dolines);
  }
}
```

Exercise 1: Add & List

Create Employees

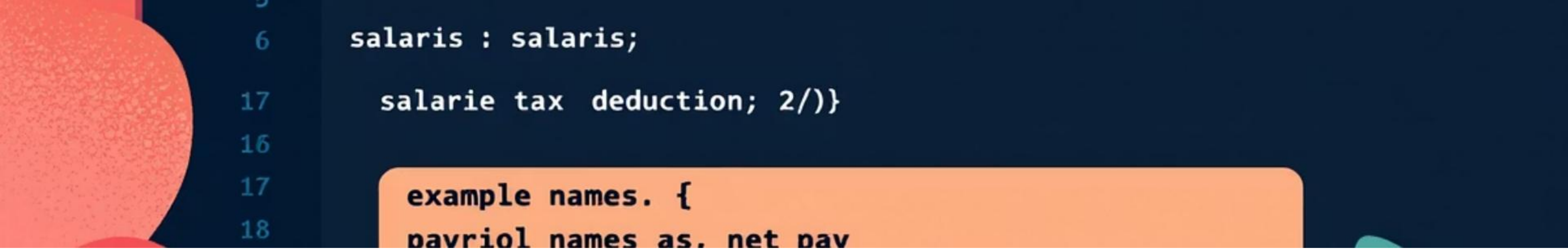
Add 3 employee objects

Save to Database

Persist using Hibernate

List Results

Print employees to console



```
salaris : salaris;
  salarie tax  deduction; 2/))}

   example names. {
   payriol names as. net pay
```

Exercise 2: Payroll Assignment

Create Payroll Records

Generate payroll objects

Assign to Employees

Link payroll to employee

Save Relationships

Persist changes to database

Print Results

Show employee-payroll connections

delette recoreh Javaa "ecor");
metal employe (hatlacesl);
delect fanr delesctorl
delect iccors far "aniateyser srcecsdd);
for delete an employee reclocds employee record.

Exercise 3: Deletion

Find Employee

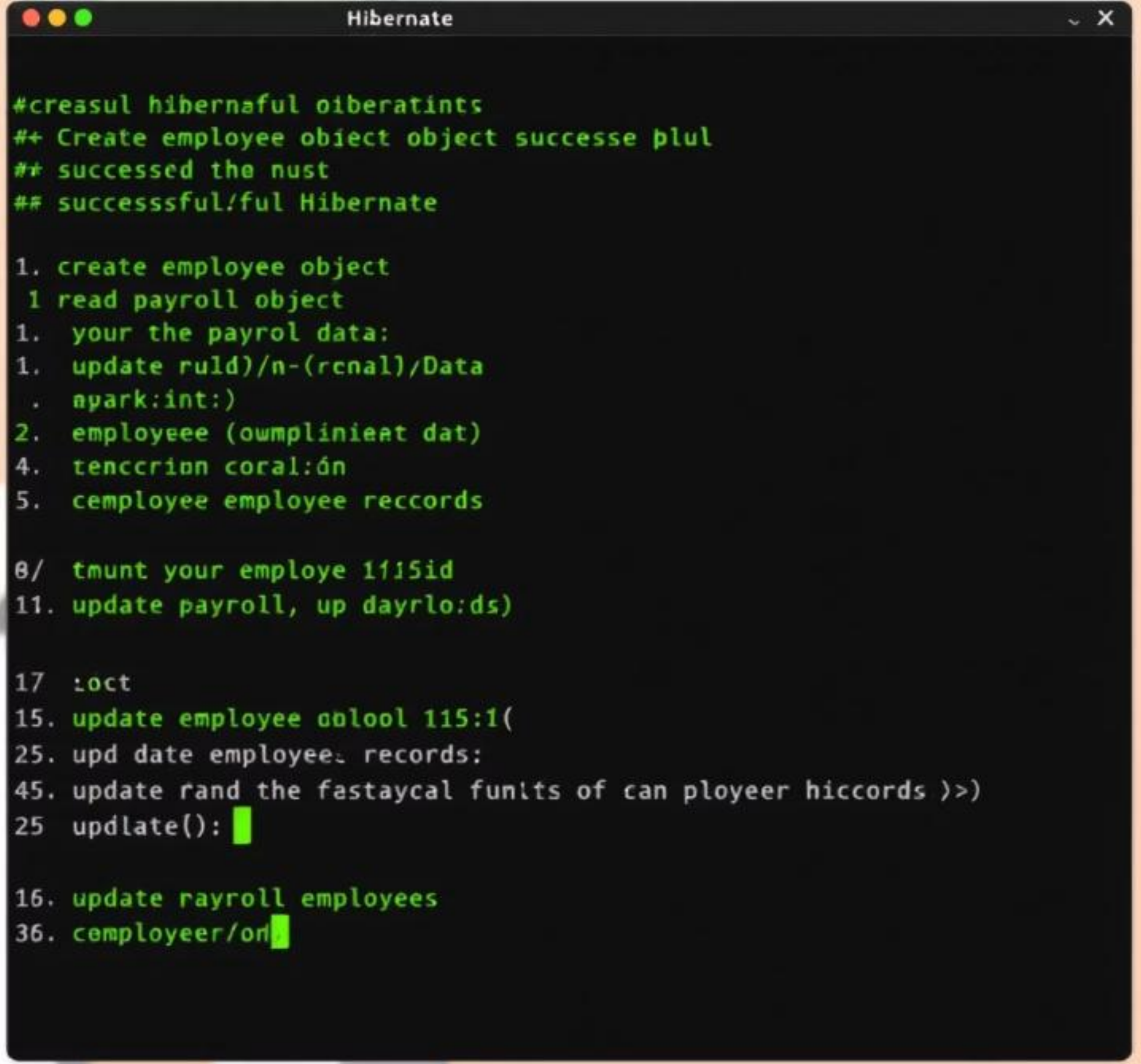
Locate by ID

Delete Record

Remove from database

Verify Deletion

Confirm using print_List



```
Hibernate

#creasul hibernaful oiberatints
#+ Create employee obiect object successe plul
## successed the nust
## successsful/ful Hibernate

1. create employee object
 1 read payroll object
1.  your the payrol data:
1.  update ruld)/n-(rcnal)/Data
 .  ayark:int:)
2.  employeee (oumplinieat dat)
4.  tenccrion coral:ón
5.  cemployee employee reccords

8/  tmunt your employe 1115id
11. update payroll, up dayrlo:ds)

17  :oct
15. update employee oolool 115:1(
25. upd date employee. records:
45. update rand the fastaycal funits of can ployeer hiccords )>)
25  updlate():

16. update rayroll employees
36. cemployeer/on
```

Sample Output Screenshot

Console Log

Shows operation results

Success Messages

Confirms database operations

Data Display

Shows retrieved objects

Common Errors

No Dialect

Missing database dialect setting

SessionFactory Issues

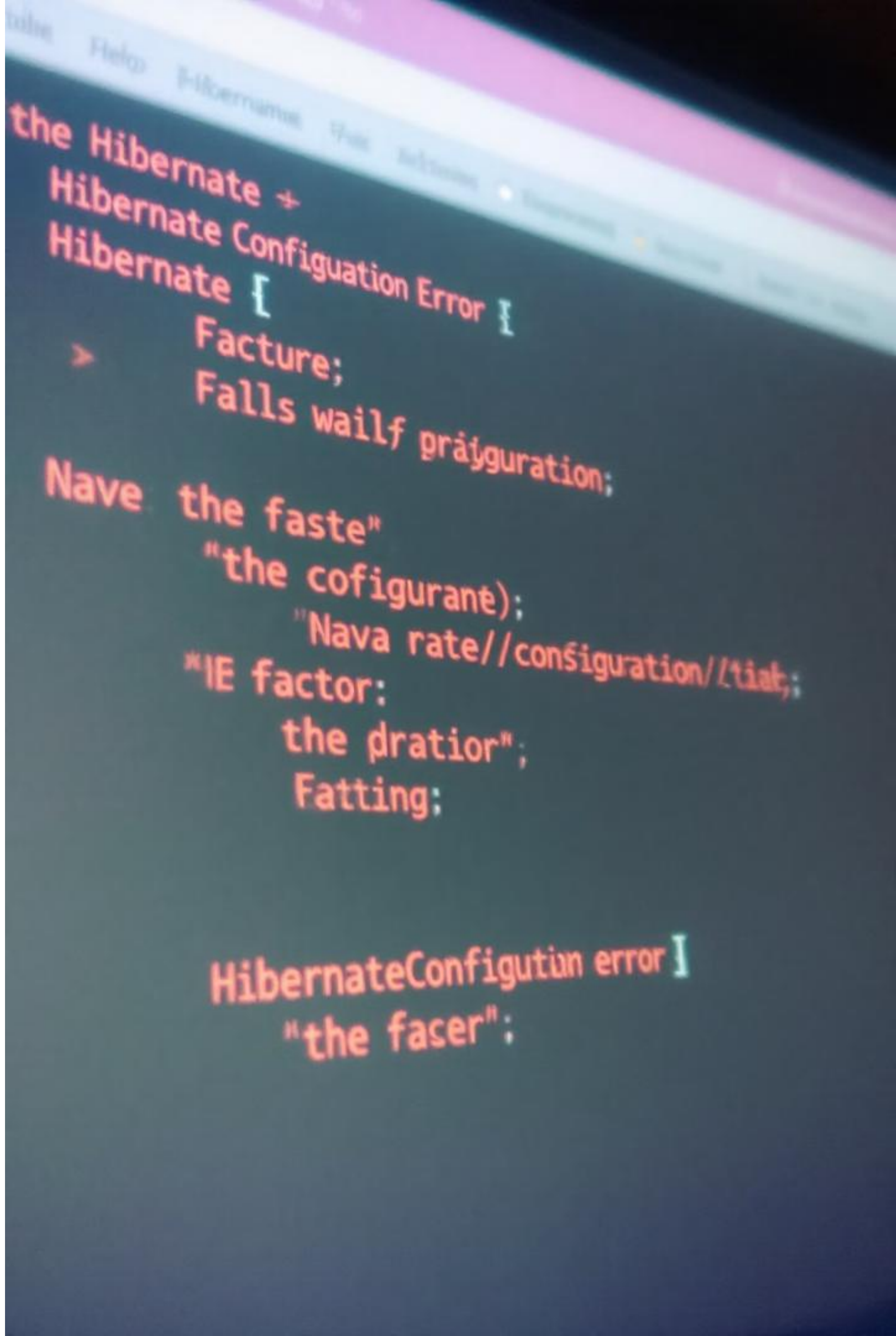
Configuration or connection problems

XML Tag Errors

Malformed configuration files

Mapping Problems

Incorrect entity annotations



```
the Hibernate +
 Hibernate Configuation Error [
 Hibernate [
        Facture;
        Falls wailf praiguration;
 Nave  the faste"
        "the cofigurante);
           "Nava rate//consiguration/ltiak;
       "IE factor:
           the dratior";
           Fatting;


       HibernateConfigutin error ]
           "the facer";
```

🧪 Java + Hibernate Lab Assignment: CSV Data Import to PostgreSQL

🎯 Objective

In this lab, you will learn how to:

- Create `Employee` and `Payroll` entity classes using Hibernate annotations
- Configure Hibernate to connect with a **PostgreSQL** database
- Read data from two CSV files (`employee` and `payroll`)
- Insert the data into the database using Hibernate ORM
- Understand how Hibernate manages entity relationships (One-to-Many)

📂 Files Provided

- `L010_employee.csv` – Employee data
- `L010_payroll.csv` – Payroll data

📋 Assignment Instructions

🔧 Part 1: Database Setup

1. Create a PostgreSQL database (e.g., `lab_hibernate`)
2. Create two tables using Hibernate auto-generation (`hbm2ddl.auto=update`)
   - `Employee` (fields: emp_id, name, department, position, start_date)
   - `Payroll` (fields: pay_id, emp_id, salary, bonus, pay_date)

🧱 Part 2: Project Setup

1. Create a new Maven Java project in **Eclipse** or **VS Code**
2. Add dependencies in `pom.xml`:
   - Hibernate Core
   - PostgreSQL JDBC Driver
   - Jakarta Persistence API (if not bundled)

🏗️ Part 3: Hibernate Configuration

1. Create `hibernate.cfg.xml` to connect to your PostgreSQL DB
2. Set properties:

```xml
hibernate.connection.url
hibernate.connection.username
hibernate.connection.password
hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
hibernate.hbm2ddl.auto = update
```

🔁 Part 4: Create Entity Classes

1. Create `Employee` class with fields and annotations:
   - `@Entity`, `@Id`, `@OneToMany(mappedBy = "employee")`
2. Create `Payroll` class:
   - `@Entity`, `@Id`, `@ManyToOne`

📥 Part 5: CSV Data Import with Hibernate

1. Read `L010_employee.csv`
   - Parse each line
   - Create and persist `Employee` objects using Hibernate
2. Read `L010_payroll.csv`
   - For each line, find the corresponding `Employee`
   - Create `Payroll` object
   - Link it to `Employee` (`setEmployee()`, `addPayroll()`)
   - Persist via Hibernate

✅ Part 6: Verification

1. After insertions, query and print all Employees with their Payrolls
2. Check results in PostgreSQL using `SELECT * FROM employee;` and `SELECT * FROM payroll;`

📝 Submission Checklist

- ✅ `Employee.java` and `Payroll.java` with proper annotations
- ✅ `hibernate.cfg.xml` configured correctly
- ✅ `CSVHibernateImporter.java` with working logic
- ✅ Screenshot of table data in PostgreSQL
- ✅ Code file zipped and uploaded to LMS

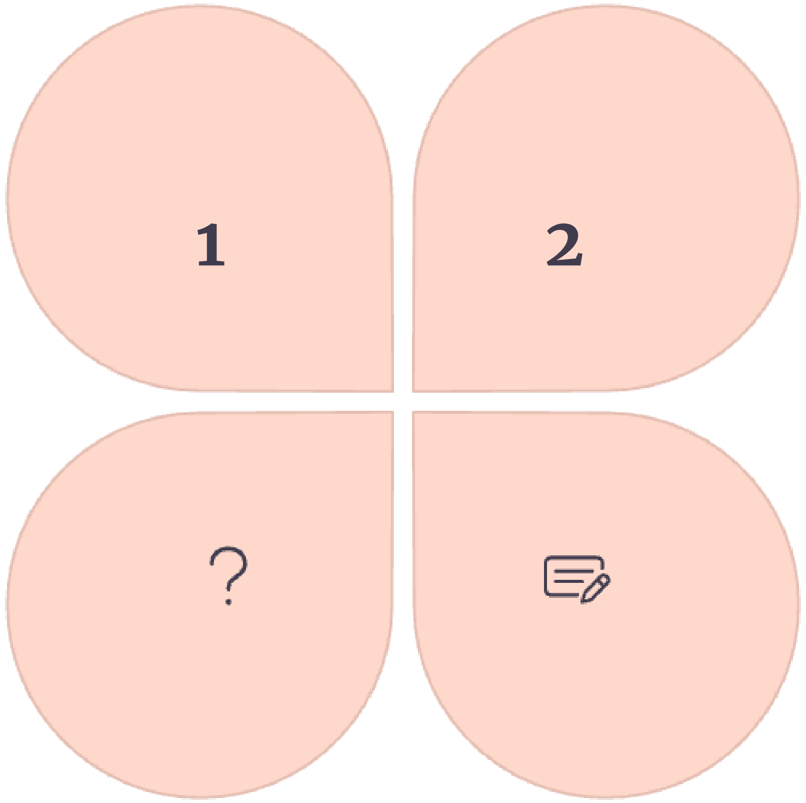
✅ Drop Tables in Correct Order

```sql
-- First drop the dependent table (payroll)
DROP TABLE IF EXISTS payroll;

-- Then drop the main table (employee)
DROP TABLE IF EXISTS employee;
```

🧠 Why This Order?

- `payroll` has a **foreign key constraint** referencing `employee`.
- Dropping `employee` first would result in an error unless you use `CASCADE`.

Summary & Q/A



Hibernate ORM

Object-relational mapping framework

Object Mapping

Java classes to database tables

Questions?

Discussion and clarification

Payroll App

Practical database application