

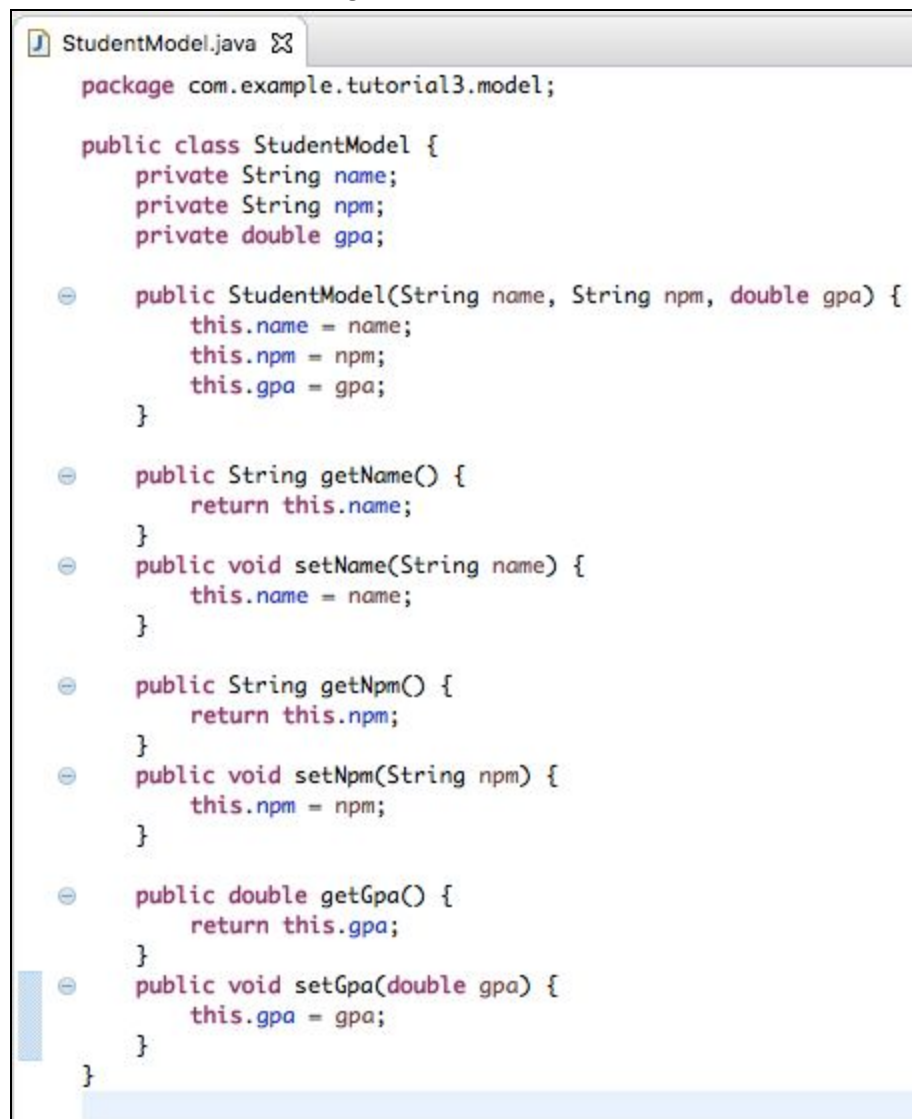
TUTORIAL 3

Ringkasan Materi

- Biasakan menggunakan getter untuk mendapatkan nilai dari sebuah model
- Perhatikan urutan saat memasukkan parameter ketika instansiasi objek, ada kemungkinan terjadi kesalahan urutan yang mengakibatkan debugging yang cukup lama
- Penjelasan setiap step dibawah ini merupakan bagian dari ringkasan materi

Membuat Class Model

Buatlah class StudentModel dengan spesifikasi seperti di atas pada package tersebut. Kemudian tambahkan method constructor, setter, dan getter.



```
StudentModel.java
package com.example.tutorial3.model;

public class StudentModel {
    private String name;
    private String npm;
    private double gpa;

    public StudentModel(String name, String npm, double gpa) {
        this.name = name;
        this.npm = npm;
        this.gpa = gpa;
    }

    public String getName() {
        return this.name;
    }

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

    public String getNpm() {
        return this.npm;
    }

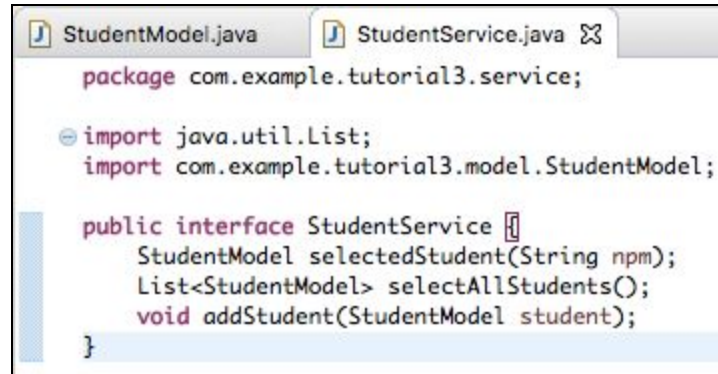
    public void setNpm(String npm) {
        this.npm = npm;
    }

    public double getGpa() {
        return this.gpa;
    }

    public void setGpa(double gpa) {
        this.gpa = gpa;
    }
}
```

Membuat Service

Buatlah interface StudentService.java pada package tersebut dengan isi sebagai berikut

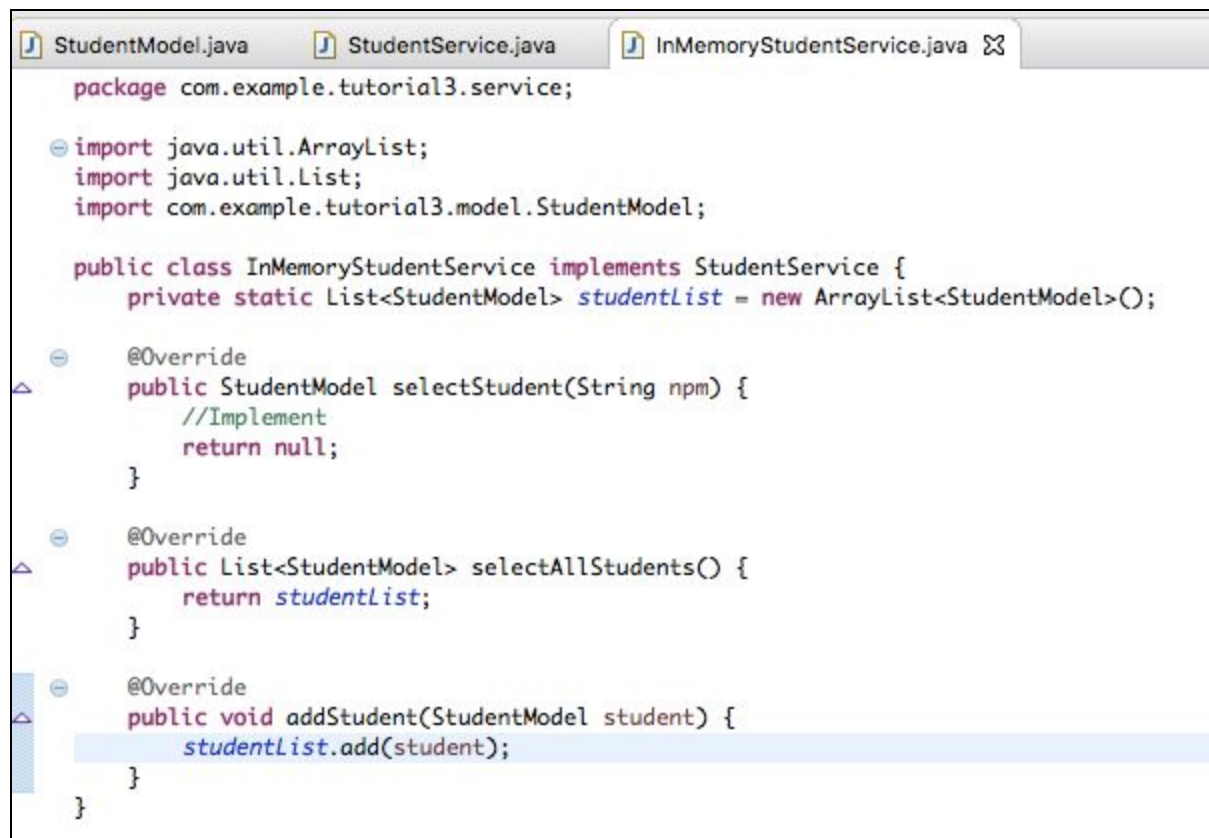


```
package com.example.tutorial3.service;

import java.util.List;
import com.example.tutorial3.model.StudentModel;

public interface StudentService {
    StudentModel selectedStudent(String npm);
    List<StudentModel> selectAllStudents();
    void addStudent(StudentModel student);
}
```

Pada package yang sama, buat class InMemoryStudentService yang meng-implements StudentService dengan isi sebagai berikut



```
package com.example.tutorial3.service;

import java.util.ArrayList;
import java.util.List;
import com.example.tutorial3.model.StudentModel;

public class InMemoryStudentService implements StudentService {
    private static List<StudentModel> studentList = new ArrayList<StudentModel>();

    @Override
    public StudentModel selectStudent(String npm) {
        //Implement
        return null;
    }

    @Override
    public List<StudentModel> selectAllStudents() {
        return studentList;
    }

    @Override
    public void addStudent(StudentModel student) {
        studentList.add(student);
    }
}
```

Implementasikan method selectStudent! Method ini menerima NPM mahasiswa dan mengembalikan object Student dengan NPM tersebut. Return null jika tidak ditemukan

```
@Override
public StudentModel selectStudent(String npm) {
    for(StudentModel student: studentList) {
        if (student.getNpm().equals(npm)) {
            return student;
        }
    }
    return null;
}
```

Membuat Controller dan Fungsi Add

Setelah membuat class StudentController dan mengimplementasi fungsi Add, kemudian menjalankan <http://localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43> , dan hasilnya tidak error. Kemudian menjalankan <http://localhost:8080/student/add?npm=12345&name=chanek> , dan hasilnya error seperti gambar dibawah



Hal ini terjadi karena parameter gpa tidak terpenuhi pada url, padahal sudah di set required=true pada request mapper.

Method View by NPM

Saya mengupdate beberapa code pada view seperti gambar dibawah ini

```
<h3 th:text="'NPM = ' + ${student.getNpm()}">Student NPM</h3>
<h3 th:text="'Name = ' + ${student.getName()}">Student NPM</h3>
<h3 th:text="'GPA = ' + ${student.getGpa()}">Student NPM</h3>
```

Hal ini dilakukan karena seharusnya data-data tersebut diakses dengan menggunakan getter, bukan langsung mengakses variable (since variable tersebut bersifat private)

Setelah mengimplementasi fungsi view, kemudian menjalankan localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/view?npm=12345

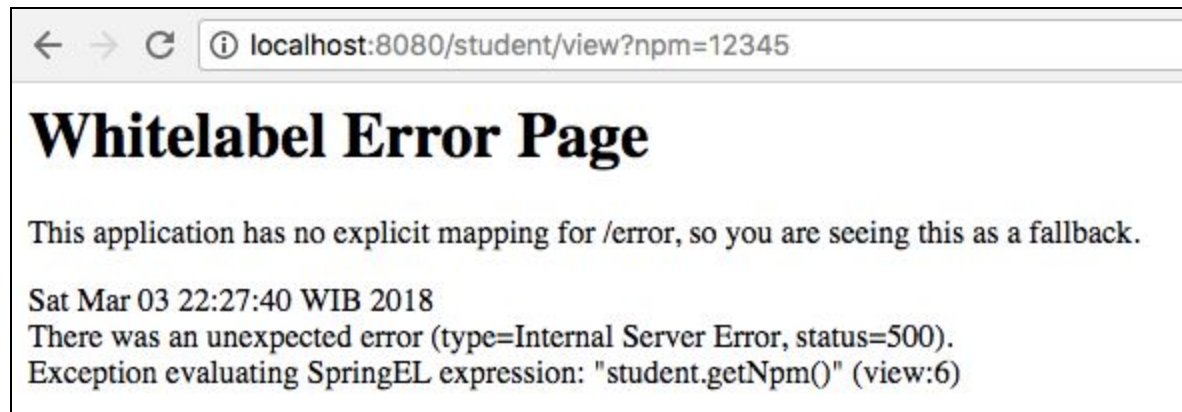
NPM = 12345

Name = chaneK

GPA = 3.43

Hasilnya adalah seperti gambar diatas

Coba matikan program dan jalankan kembali serta buka localhost:8080/student/view?npm=12345



Hasilnya seperti error diatas, instruksi yang dilakukan hanya merestart server dan mengakses view, tanpa menambah student terlebih dahulu. Server menyimpan data tidak di persistent storage melainkan pada memory, jadi ketika server di restart, data yang sudah ditambahkan sebelumnya (yang ada pada memory) akan hilang.

Method View All

Setelah mengimplementasi fungsi view, kemudian menjalankan localhost:8080/student/add?npm=12345&name=chaneK&gpa=3.43 lalu buka localhost:8080/student/view?npm=12345
Apakah data Student tersebut muncul? Ya data student tersebut muncul
Setelah menambahkan data student lain, Apakah semua data Student muncul?
Ya semua data student muncul

Latihan 1

Langkah pertama, yaitu menambahkan method viewStudentWithNpm seperti gambar dibawah ini

```
@RequestMapping("/student/view/{npm}")
public String viewStudentWithNpm(Model model, @PathVariable(required = false) Optional<String> npm) {
    if(npm.isPresent()) {
        StudentModel student = studentService.selectStudent(npm.get());
        if(student != null) {
            model.addAttribute("student", student);
            return "view";
        } else {
            model.addAttribute("errorMessage", "Student with NPM : "+npm.get()+" not found");
            return "errorPage";
        }
    } else {
        model.addAttribute("errorMessage", "Please insert NPM!");
        return "errorPage";
    }
}
```

Melakukan pengecekan pada parameter apakah exist or not, jika tidak maka masuk ke page error dengan error message yang di hardcode.

Jika ya, maka melakukan pencarian terhadap npm. Jika ditemukan, maka menampilkan data student.

Jika tidak maka akan menampilkan error message yang telah ditambahkan NPM yang menyatakan murid dengan npm tersebut tidak ditemukan.

Kemudian menambahkan errorPage seperti gambar berikut

```
<!DOCTYPE html>
<html xmlns:th = "http://www.thymeleaf.org">
    <head>
        <title> Error 404 </title>
    </head>
    <body>
        <h3 th:text = "${errorMessage}" > Error Message </h3>
    </body>
</html>
```

Simpelnya, halaman ini menampilkan string error message yang telah di racik pada method viewStudentWithNpm.

Latihan 2

Langkah pertama, yaitu menambahkan method viewStudentWithNpm seperti gambar dibawah ini

```
@RequestMapping("/student/view/{npm}")
public String viewStudentWithNpm(Model model, @PathVariable(required = false) Optional<String> npm) {
    if(npm.isPresent()) {
        StudentModel student = studentService.selectStudent(npm.get());
        if(student != null) {
            model.addAttribute("student", student);
            return "view";
        } else {
            model.addAttribute("errorMessage", "Student with NPM : "+npm.get()+" not found");
            return "errorPage";
        }
    } else {
        model.addAttribute("errorMessage", "Please insert NPM!");
        return "errorPage";
    }
}
```

Melakukan pengecekan pada parameter apakah exist or not, jika tidak maka masuk ke page error dengan error message yang di hardcode.

Jika ya, maka melakukan pencarian terhadap npm. Jika ditemukan, maka akan menghapus data student. Jika tidak maka akan menampilkan error message yang telah ditambahkan NPM yang menyatakan murid dengan npm tersebut tidak ditemukan dan penghapusan dibatalkan.

Kemudian menambahkan successPage seperti gambar berikut

```
<!DOCTYPE html>
<html xmlns:th = "http://www.thymeleaf.org">
<head>
<title> Delete success </title>
</head>
<body>
<h3 th:text = "${message}" > Message </h3>
</body>
</html>
```

Page error tidak dibikin kembali karena bisa memakai yang sudah dibuat pada latihan sebelumnya.