Membuat Project

Membuat Project

- https://start.spring.io/
- Java 21

Sealed Class

Sealed Class

- Fitur Sealed Class pertama kali diperkenalkan pada versi Java 15, setelah terjadi beberapa perubahan, akhirnya rilis final pada versi Java 17
- https://openjdk.org/jeps/409
- Sealed class atau interface, adalah class atau interface yang hanya bisa diturunkan oleh class atau interface yang diperbolehkan saja

Tujuan Sealed Class

- Memberikan pembuat class atau interface bisa mengontrol, kode mana yang bisa extends atau implementasi kode yang sudah dibuat
- Lebih deklaratif mendeskripsikan siapa yang bisa mengakses dibanding menggunakan access modifier yang terbatas
- Bisa digunakan untuk fitur baru Java bernama Pattern Matching
- Sealed class bukanlah final class, final class tidak bisa diturunkan ke class apapun, sedangkan sealed class bisa diturunkan, hanya saja pada class yang kita pilih saja

Kode : Sealed Class

```
public sealed class Employee {

  private String id;

  private String name;

  public String getId() {
    return id;
  }
```

Minimum Sub Class

- Saat kita membuat Sealed Class, minimal harus ada satu Sub Class, karena jika kita membuat Sealed Class yang tidak memiliki Sub Class, artinya sama saja seperti final Class
- Oleh karena itu, kita wajib menambahkan minimal satu Sub Class
- Ketika menambahkan Sub Class, kita wajib memberi tahu Sub Class mana yang boleh melakukan extends / implements ke Sealed Class menggunakan kata kunci permits
- Sub Class wajib Sealed Class lagi atau final Class

Kode : Sealed Class

```
public sealed class Employee permits Manager {

  private String id;

  private String name;

  public String getId() {
    return id;
  }
```

Kode : Sub Class

```
public final class Manager extends Employee {

  private String department;

  public String getDepartment() {
    return department;
  }
```

Non Sealed Class

Non Sealed Class

- Seperti di materi sebelumnya dijelaskan, saat membuat Sub Class dari Sealed Class, kita wajib menggunakan Sealed Class atau final Class
- Jika memang kita ingin membuat Sub Class yang harapannya bisa di extends lagi oleh Sub Class lain, maka kita bisa menggunakan Sealed Class, namun secara otomatis wajib menyebutkan Sub Class yang di permits nya lagi
- Atau kita bisa gunakan Non Sealed Class, menggunakan kata kunci non-sealed

Kode : Non Sealed Class

```
public non-sealed class Manager extends Employee {

  private String department;

  public String getDepartment() {
    return department;
  }
```

Sealed Interface

Sealed Interface

- Seperti dijelaskan di awal, Sealed Class tidak hanya bisa digunakan di Class, namun bisa juga di Interface
- Cara nya sama, ketika kita membuat Sealed Interface, kita harus sebutkan menggunakan permits, interface mana yang bisa extends, atau class mana yang bisa implements
- Dan jika menggunakan Class, kita juga harus pastikan bahwa class nya adalah Sealed Class, final Class atau Non Sealed Class

Kode : Sealed Interface

```
public sealed interface Shape permits Circle, Rectangle, Triangle {

  Long area();

}
```

Record

Record

- Record adalah salah satu implementasi dari Sub Class yang baik untuk Sealed Class
- Hal ini karena Record itu sifatnya adalah final Class, artinya memang tidak bisa diturunkan lagi
- Sehingga sangat cocok ketika menggunakan Sealed Class, lalu implementasinya adalah Record
- Namun karena Record itu tidak bisa extends ke Class, jadi Record hanya bisa implements ke Sealed Interface

Kode : Sealed Interface

```
public sealed interface SayHello permits Human, Dog, Cat{

  String hello();

}
```

Kode : Record


public record Human() implements SayHello {
@Override
public String hello() {
return "hello";
}
}

Reflection

Reflection

- Saat kita menggunakan Reflection, kita juga bisa mendapat informasi seputar Sealed Class di java.lang.Class
- Kita bisa menggunakan method isSealed() untuk mendapatkan informasi apakah ini Sealed Class atau bukan
- Dan kita bisa menggunakan getPermittedSubclasses() untuk mendapatkan informasi seluruh Sub Class dari Sealed Class

Kode : Reflection

```java
@Test
void reflection() {
  assertTrue(SayHello.class.isSealed());

  Class<?>[] subclasses = SayHello.class.getPermittedSubclasses();
  assertEquals(3, subclasses.length);
  assertEquals(Human.class, subclasses[0]);
  assertEquals(Dog.class, subclasses[1]);
  assertEquals(Cat.class, subclasses[2]);
}
```