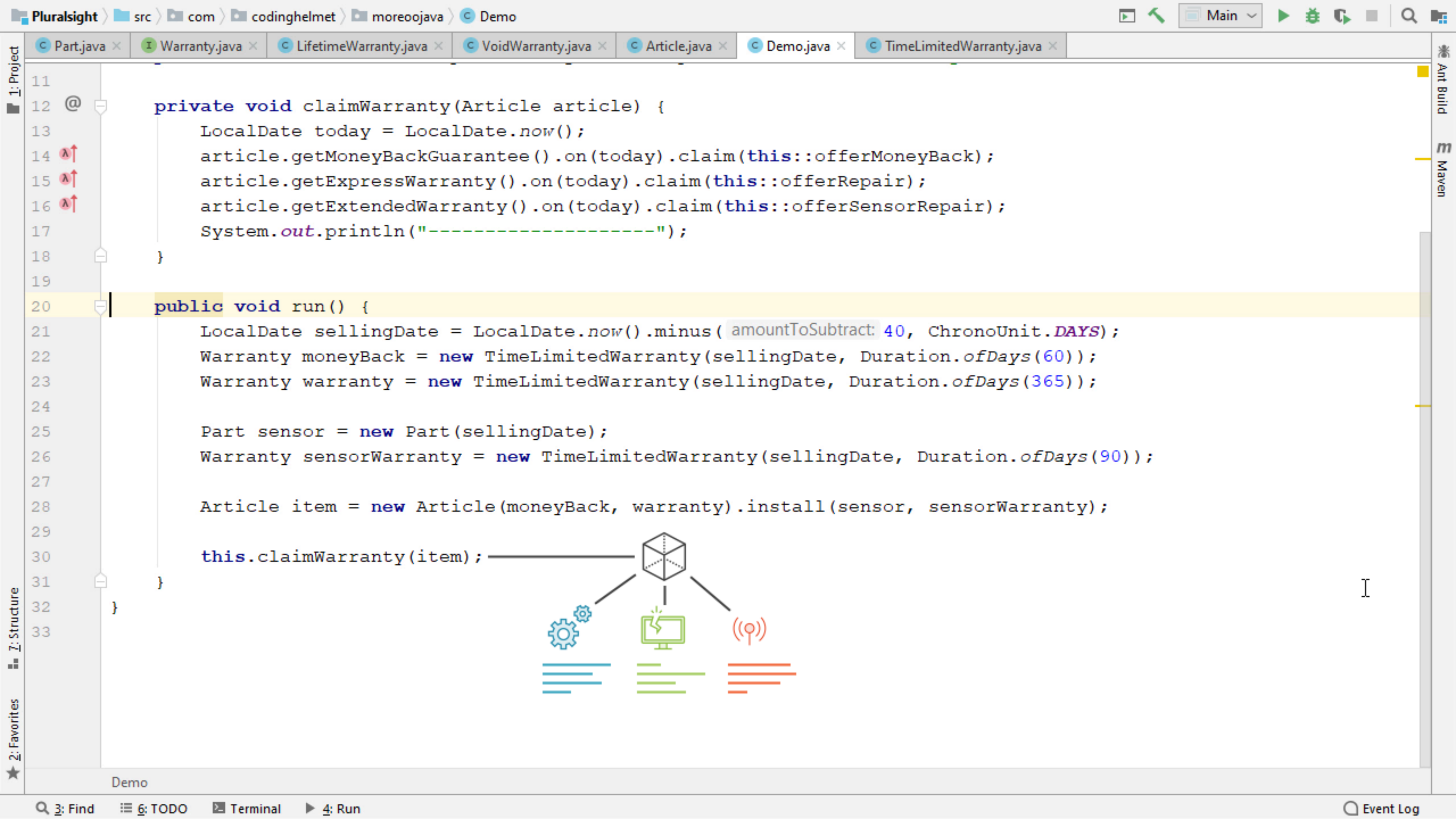
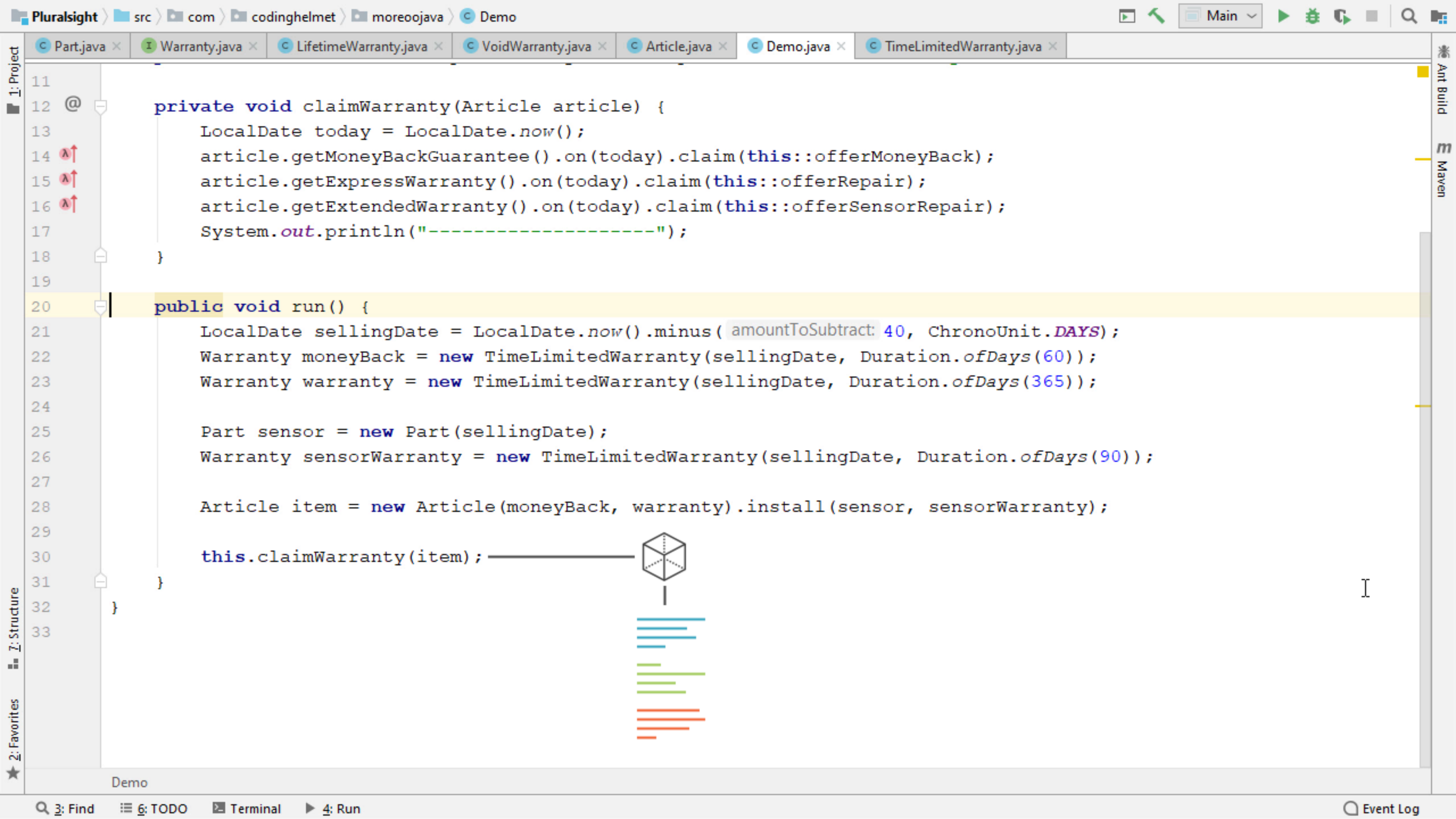
Increasing Flexibility by Removing Enums and Switch Statements

Zoran Horvat
CEO AT CODING HELMET

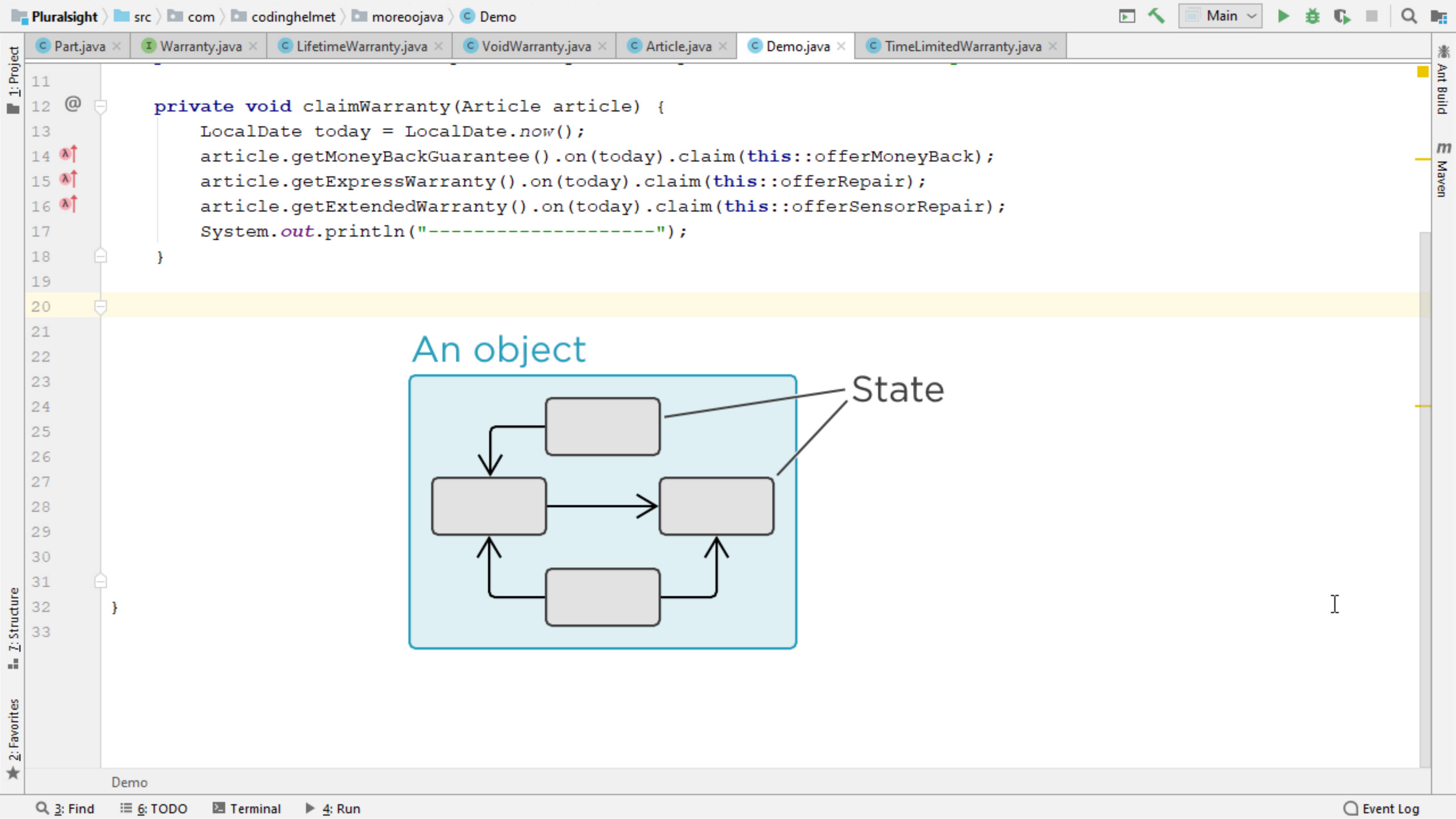
@zoranh75   http://codinghelmet.com



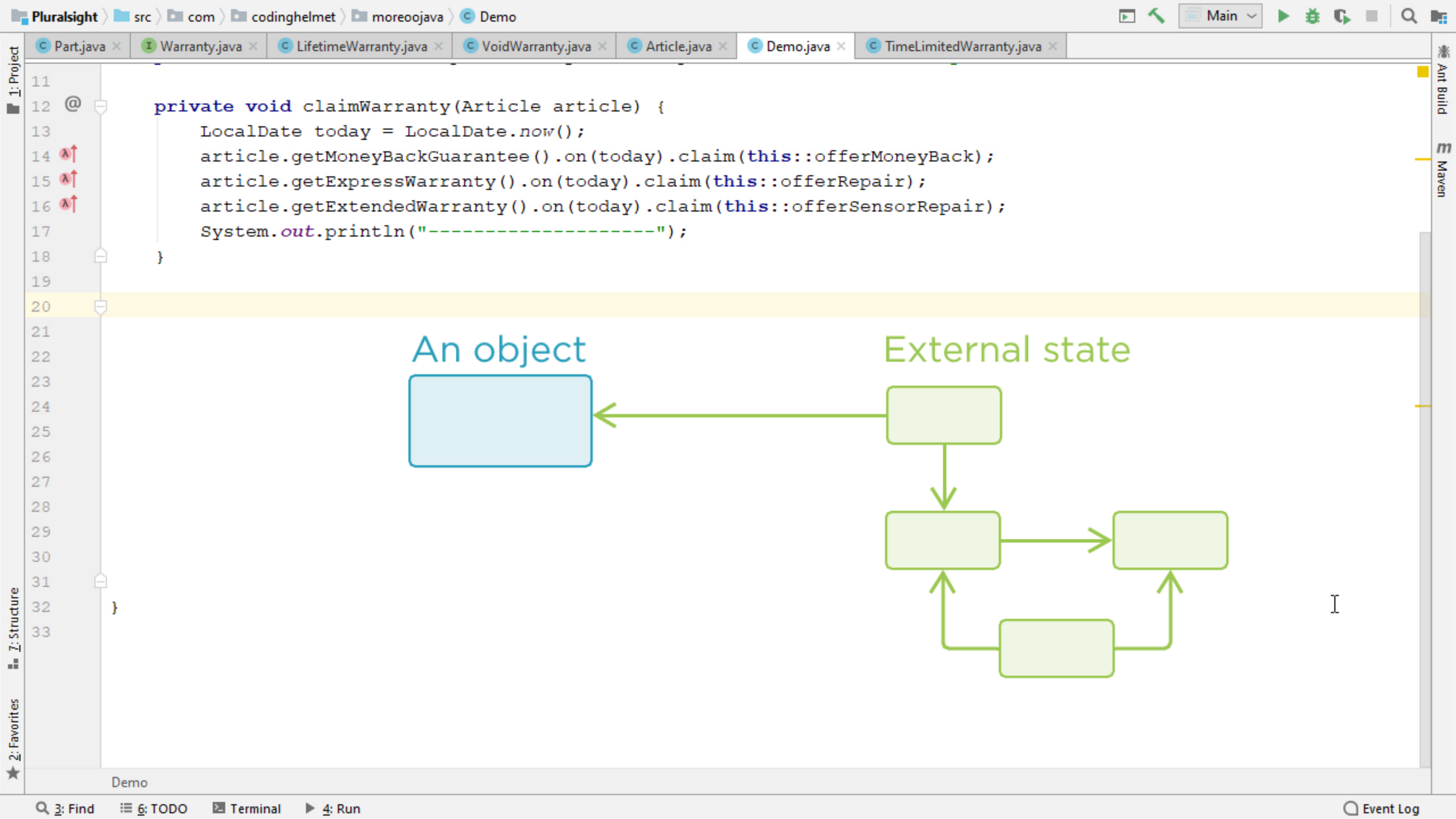
```java

    private void claimWarranty(Article article) {
        LocalDate today = LocalDate.now();
        article.getMoneyBackGuarantee().on(today).claim(this::offerMoneyBack);
        article.getExpressWarranty().on(today).claim(this::offerRepair);
        article.getExtendedWarranty().on(today).claim(this::offerSensorRepair);
        System.out.println("--------------------");
    }

    public void run() {
        LocalDate sellingDate = LocalDate.now().minus( amountToSubtract: 40, ChronoUnit.DAYS);
        Warranty moneyBack = new TimeLimitedWarranty(sellingDate, Duration.ofDays(60));
        Warranty warranty = new TimeLimitedWarranty(sellingDate, Duration.ofDays(365));

        Part sensor = new Part(sellingDate);
        Warranty sensorWarranty = new TimeLimitedWarranty(sellingDate, Duration.ofDays(90));

        Article item = new Article(moneyBack, warranty).install(sensor, sensorWarranty);

        this.claimWarranty(item);
    }
}

```



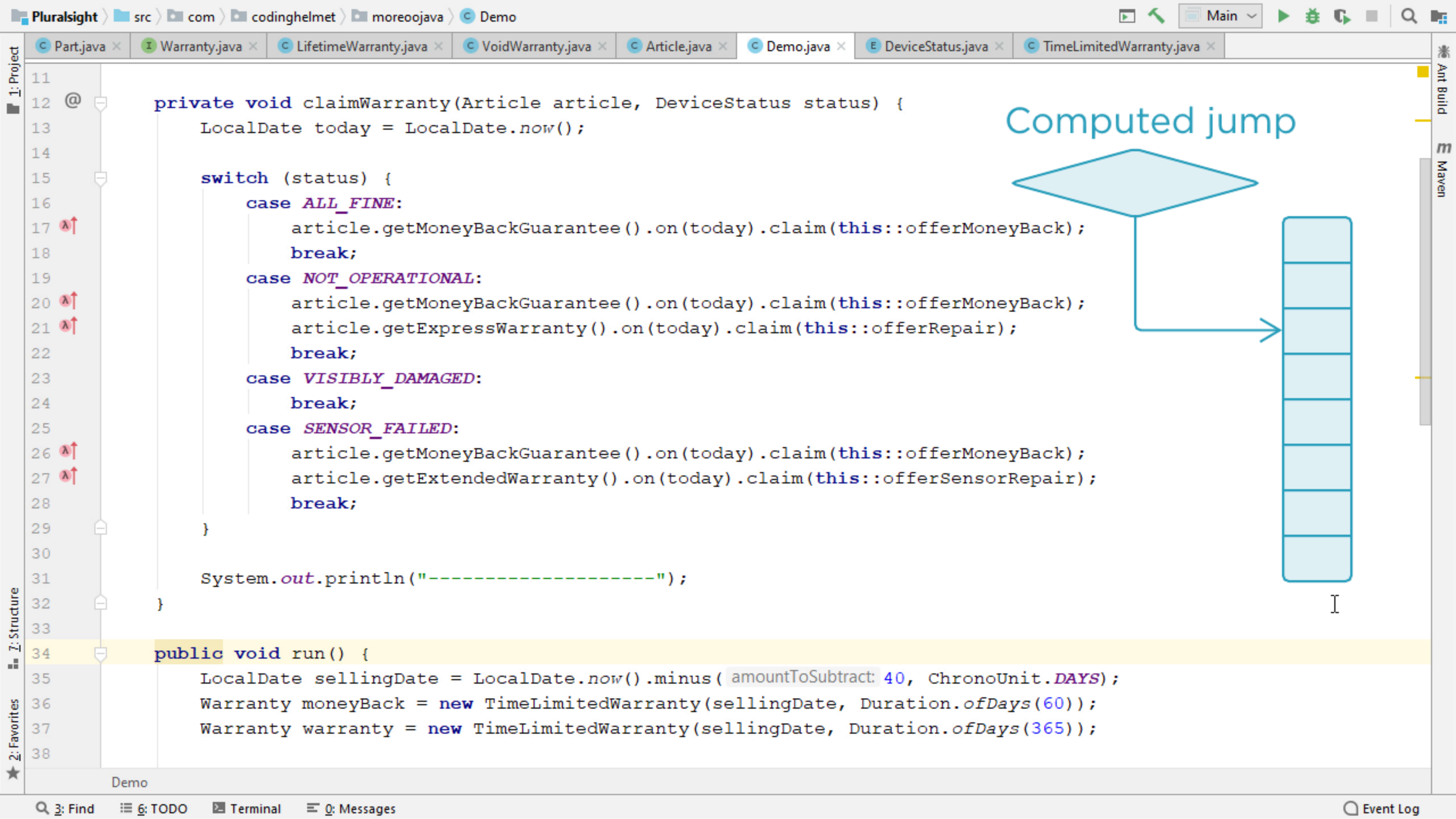
```java

    private void claimWarranty(Article article) {
        LocalDate today = LocalDate.now();
        article.getMoneyBackGuarantee().on(today).claim(this::offerMoneyBack);
        article.getExpressWarranty().on(today).claim(this::offerRepair);
        article.getExtendedWarranty().on(today).claim(this::offerSensorRepair);
        System.out.println("-------------------");
    }

    public void run() {
        LocalDate sellingDate = LocalDate.now().minus( amountToSubtract: 40, ChronoUnit.DAYS);
        Warranty moneyBack = new TimeLimitedWarranty(sellingDate, Duration.ofDays(60));
        Warranty warranty = new TimeLimitedWarranty(sellingDate, Duration.ofDays(365));

        Part sensor = new Part(sellingDate);
        Warranty sensorWarranty = new TimeLimitedWarranty(sellingDate, Duration.ofDays(90));

        Article item = new Article(moneyBack, warranty).install(sensor, sensorWarranty);

        this.claimWarranty(item);
    }
}

```



```java

    private void claimWarranty(Article article) {
        LocalDate today = LocalDate.now();
        article.getMoneyBackGuarantee().on(today).claim(this::offerMoneyBack);
        article.getExpressWarranty().on(today).claim(this::offerRepair);
        article.getExtendedWarranty().on(today).claim(this::offerSensorRepair);
        System.out.println("-------------------");
    }













}

```

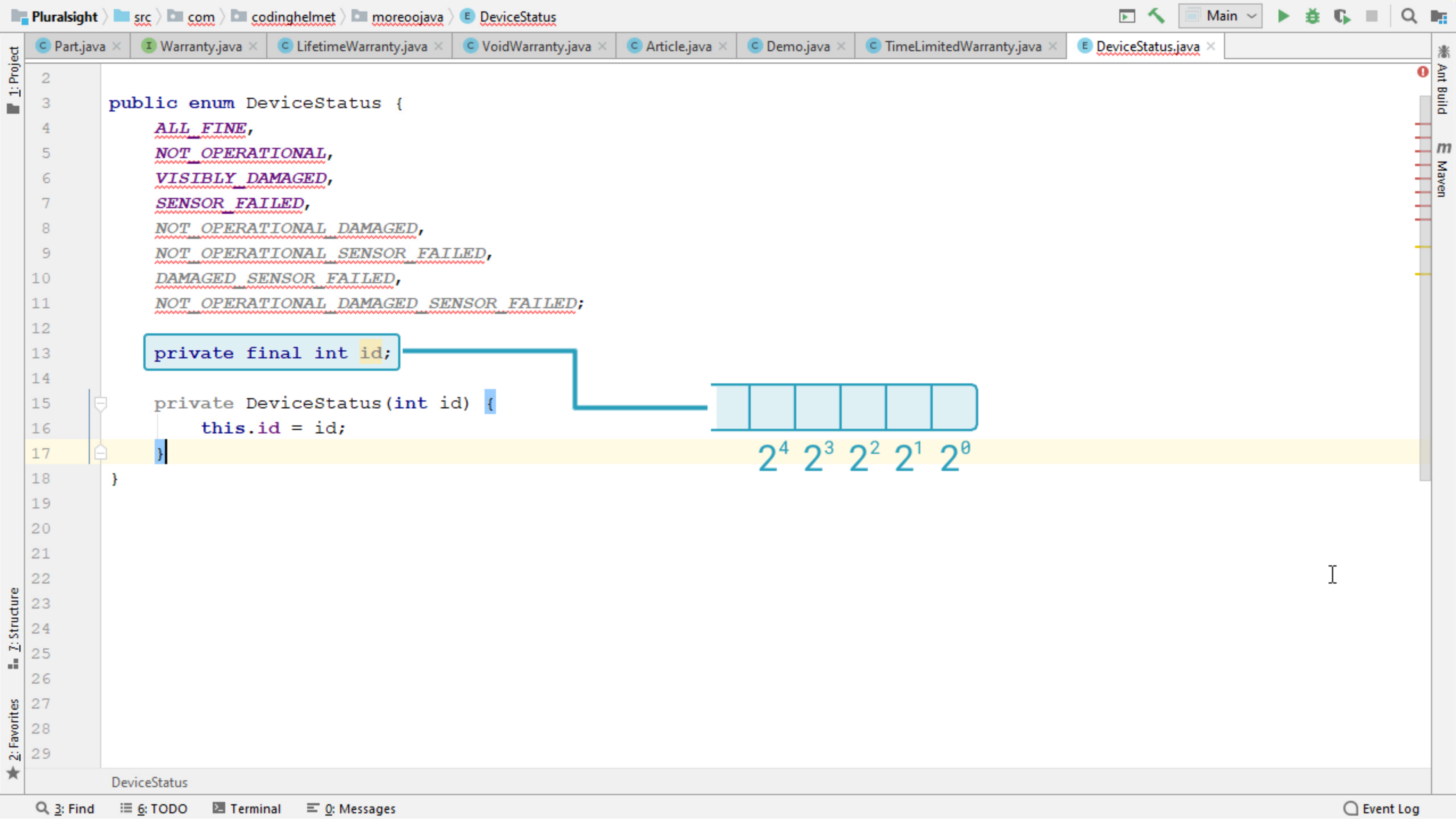


```java

    private void claimWarranty(Article article) {
        LocalDate today = LocalDate.now();
        article.getMoneyBackGuarantee().on(today).claim(this::offerMoneyBack);
        article.getExpressWarranty().on(today).claim(this::offerRepair);
        article.getExtendedWarranty().on(today).claim(this::offerSensorRepair);
        System.out.println("-------------------");
    }















}

```

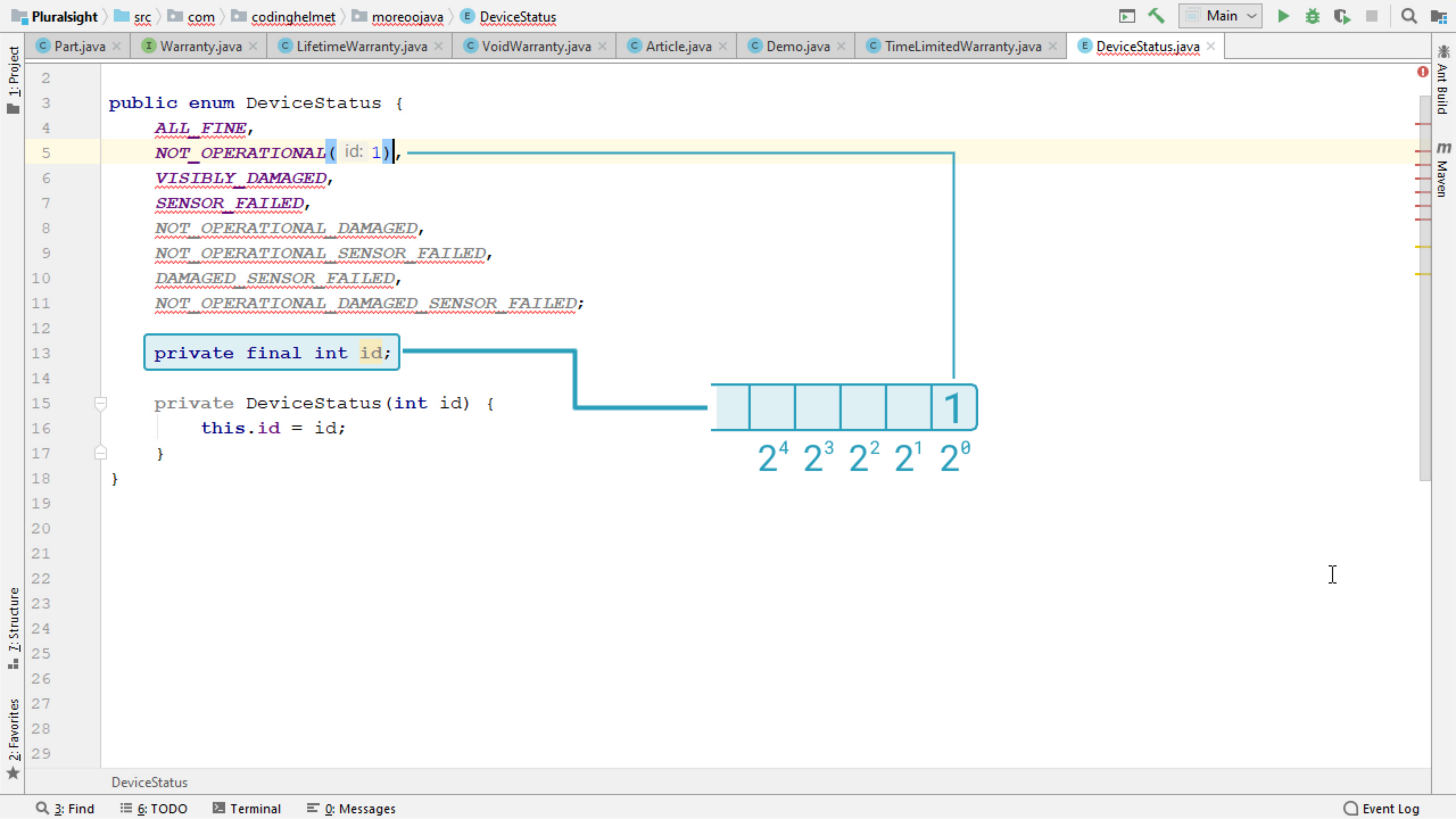


Computed jump

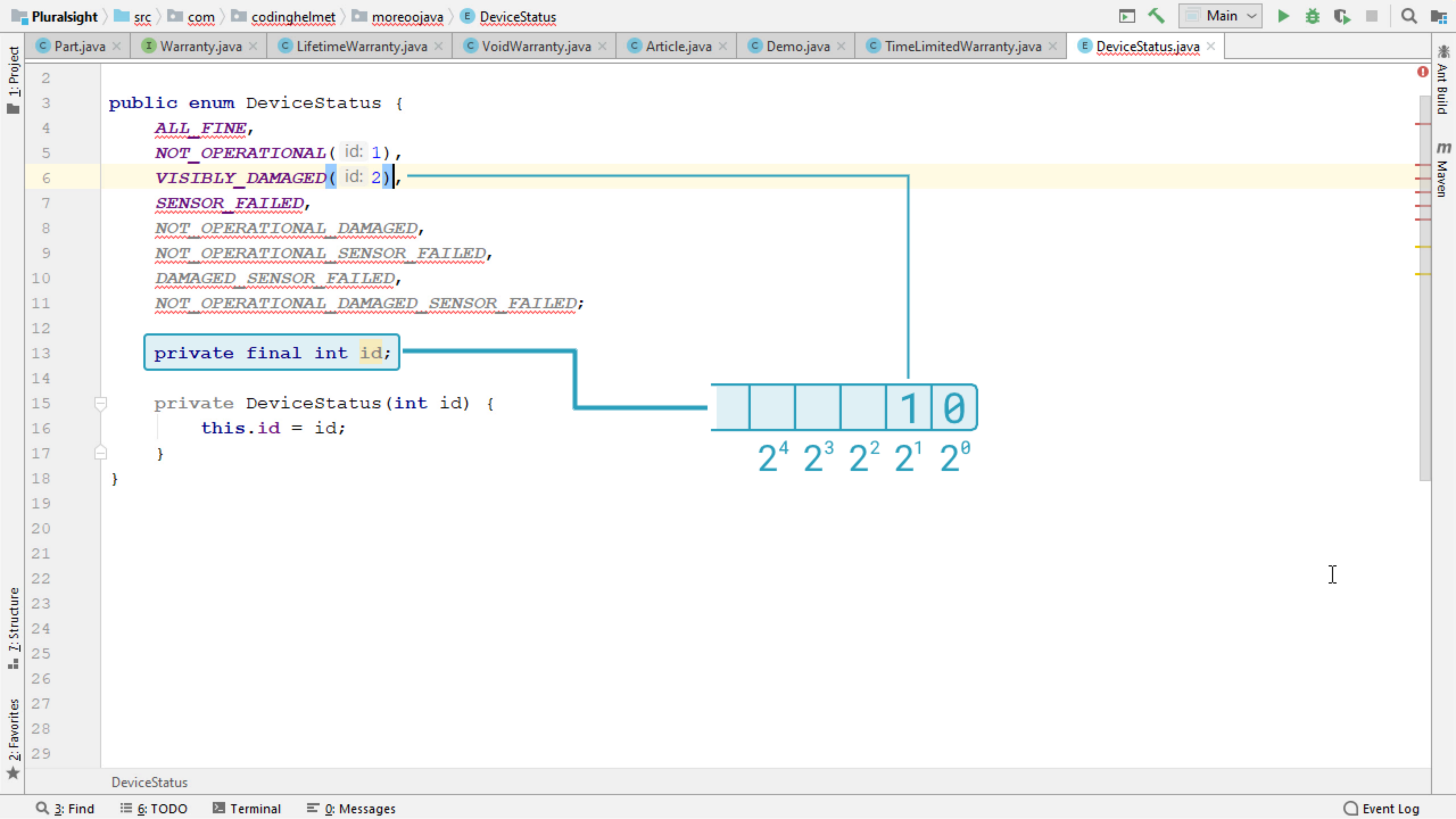
```java

    private void claimWarranty(Article article, DeviceStatus status) {
        LocalDate today = LocalDate.now();

        switch (status) {
            case ALL_FINE:
                article.getMoneyBackGuarantee().on(today).claim(this::offerMoneyBack);
                break;
            case NOT_OPERATIONAL:
                article.getMoneyBackGuarantee().on(today).claim(this::offerMoneyBack);
                article.getExpressWarranty().on(today).claim(this::offerRepair);
                break;
            case VISIBLY_DAMAGED:
                break;
            case SENSOR_FAILED:
                article.getMoneyBackGuarantee().on(today).claim(this::offerMoneyBack);
                article.getExtendedWarranty().on(today).claim(this::offerSensorRepair);
                break;
        }

        System.out.println("--------------------");
    }

    public void run() {
        LocalDate sellingDate = LocalDate.now().minus( amountToSubtract: 40, ChronoUnit.DAYS);
        Warranty moneyBack = new TimeLimitedWarranty(sellingDate, Duration.ofDays(60));
        Warranty warranty = new TimeLimitedWarranty(sellingDate, Duration.ofDays(365));

```



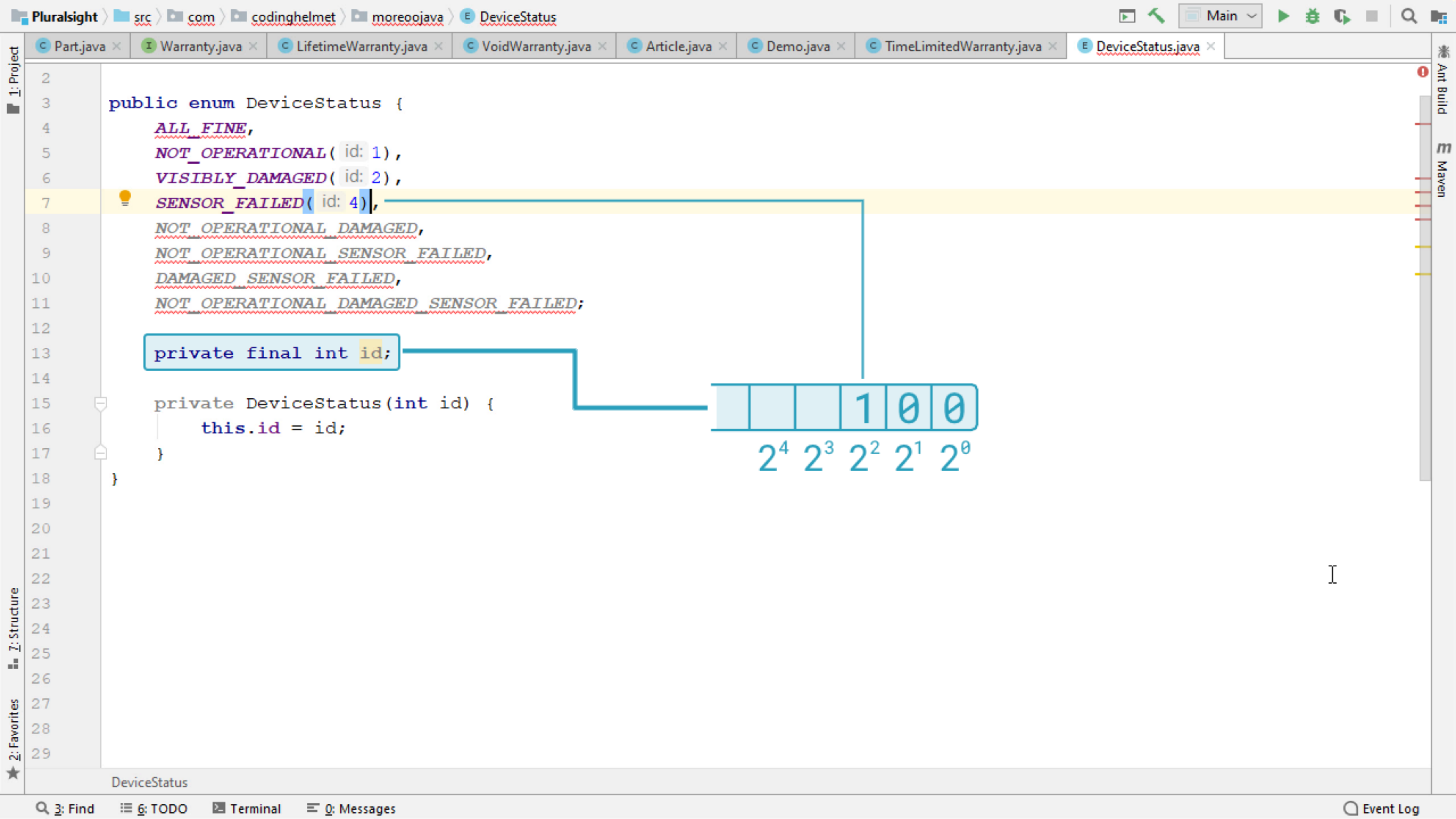
```java

public enum DeviceStatus {
    ALL_FINE,
    NOT_OPERATIONAL,
    VISIBLY_DAMAGED,
    SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED,
    NOT_OPERATIONAL_SENSOR_FAILED,
    DAMAGED_SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED_SENSOR_FAILED;

    private final int id;

    private DeviceStatus(int id) {
        this.id = id;
    }
}
```



```
public enum DeviceStatus {
    ALL_FINE,
    NOT_OPERATIONAL( id: 1),
    VISIBLY_DAMAGED,
    SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED,
    NOT_OPERATIONAL_SENSOR_FAILED,
    DAMAGED_SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED_SENSOR_FAILED;

    private final int id;

    private DeviceStatus(int id) {
        this.id = id;
    }
}
```

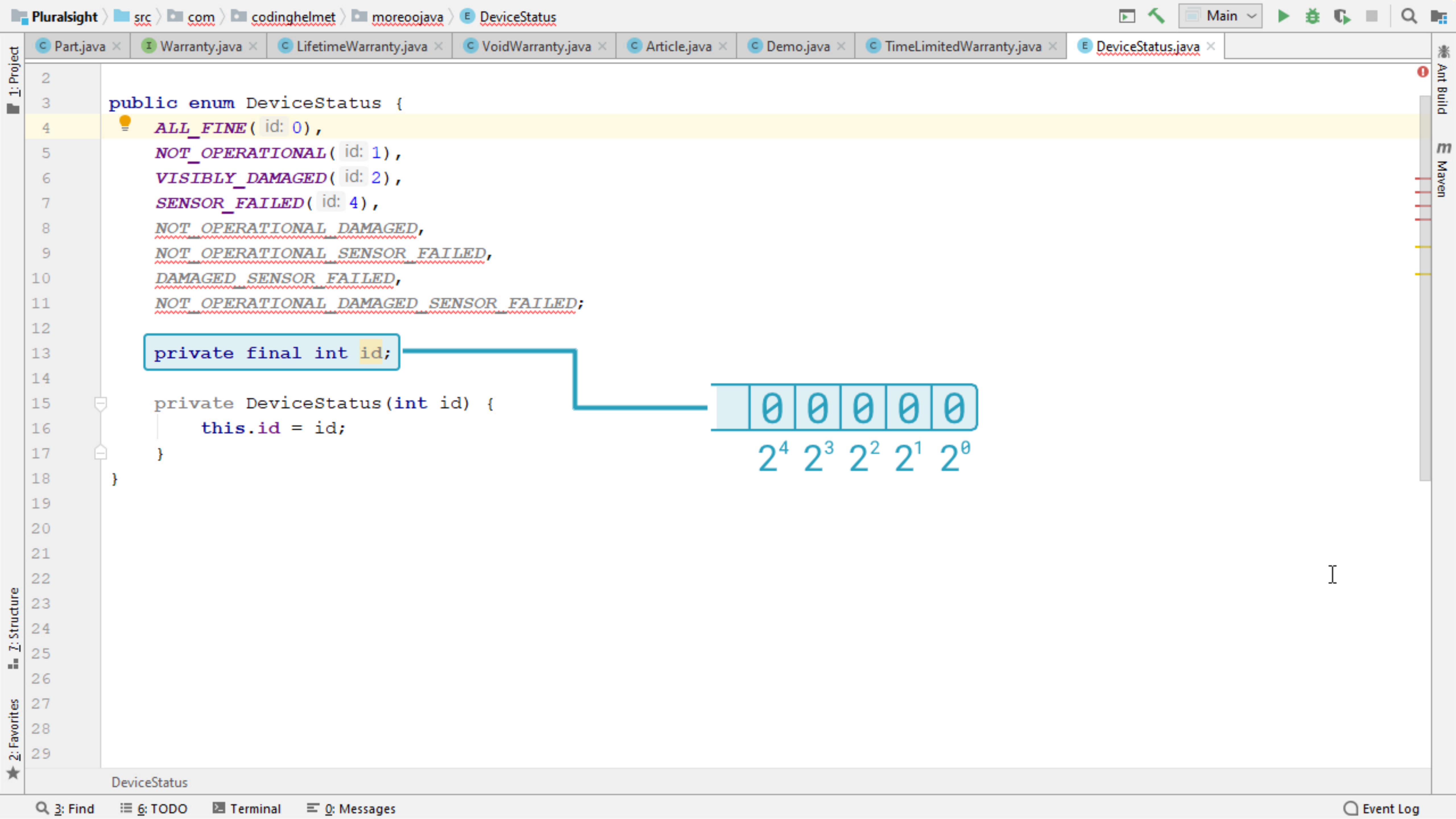


```java

public enum DeviceStatus {
    ALL_FINE,
    NOT_OPERATIONAL( id: 1),
    VISIBLY_DAMAGED( id: 2),
    SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED,
    NOT_OPERATIONAL_SENSOR_FAILED,
    DAMAGED_SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED_SENSOR_FAILED;

    private final int id;

    private DeviceStatus(int id) {
        this.id = id;
    }
}
```

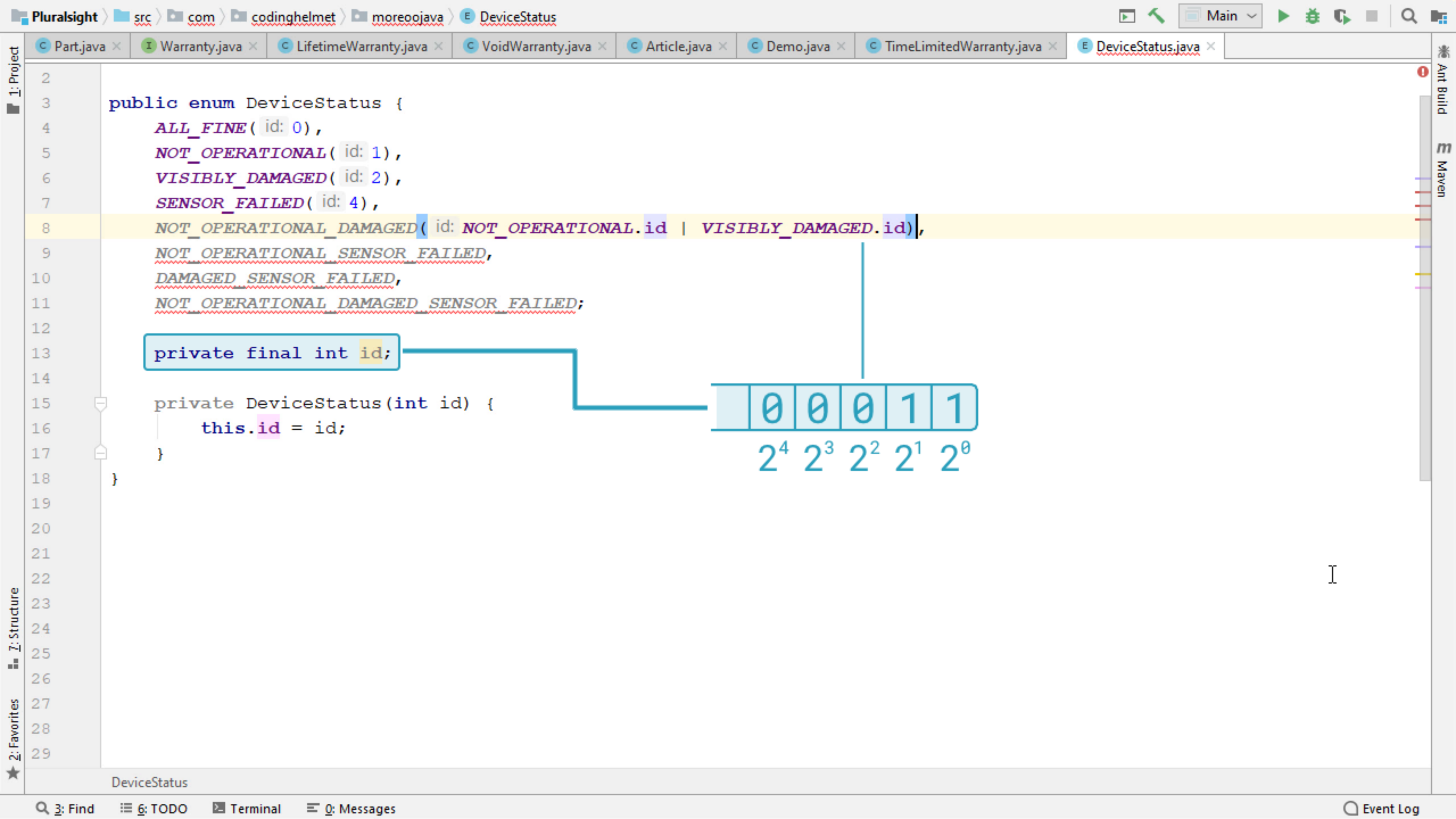


```

public enum DeviceStatus {
    ALL_FINE,
    NOT_OPERATIONAL( id: 1),
    VISIBLY_DAMAGED( id: 2),
    SENSOR_FAILED( id: 4),
    NOT_OPERATIONAL_DAMAGED,
    NOT_OPERATIONAL_SENSOR_FAILED,
    DAMAGED_SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED_SENSOR_FAILED;

    private final int id;

    private DeviceStatus(int id) {
        this.id = id;
    }
}
```

		1	0	0
2^4	2^3	2^2	2^1	2^0

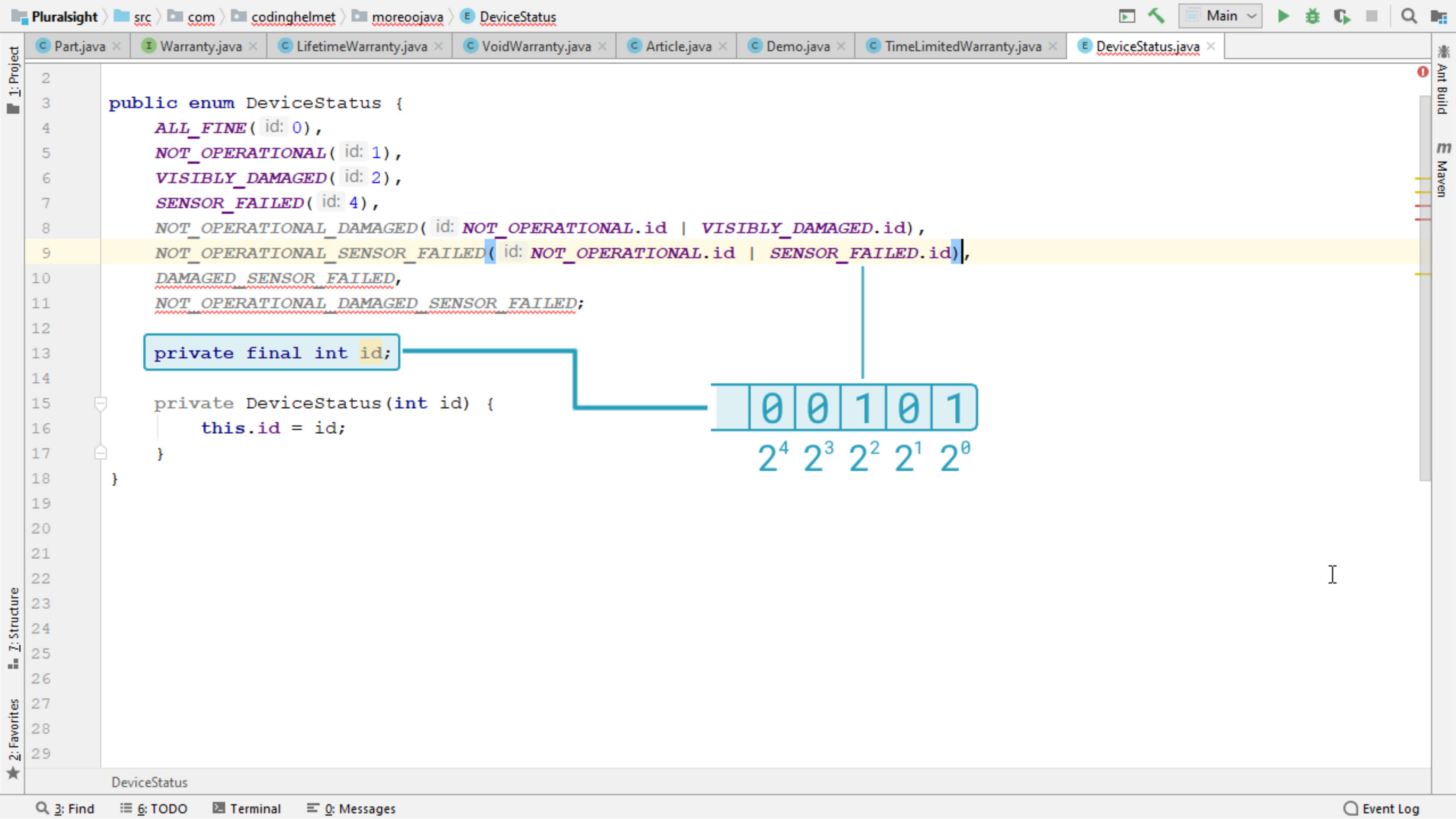


```java

public enum DeviceStatus {
    ALL_FINE( id: 0),
    NOT_OPERATIONAL( id: 1),
    VISIBLY_DAMAGED( id: 2),
    SENSOR_FAILED( id: 4),
    NOT_OPERATIONAL_DAMAGED,
    NOT_OPERATIONAL_SENSOR_FAILED,
    DAMAGED_SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED_SENSOR_FAILED;

    private final int id;

    private DeviceStatus(int id) {
        this.id = id;
    }
}
```

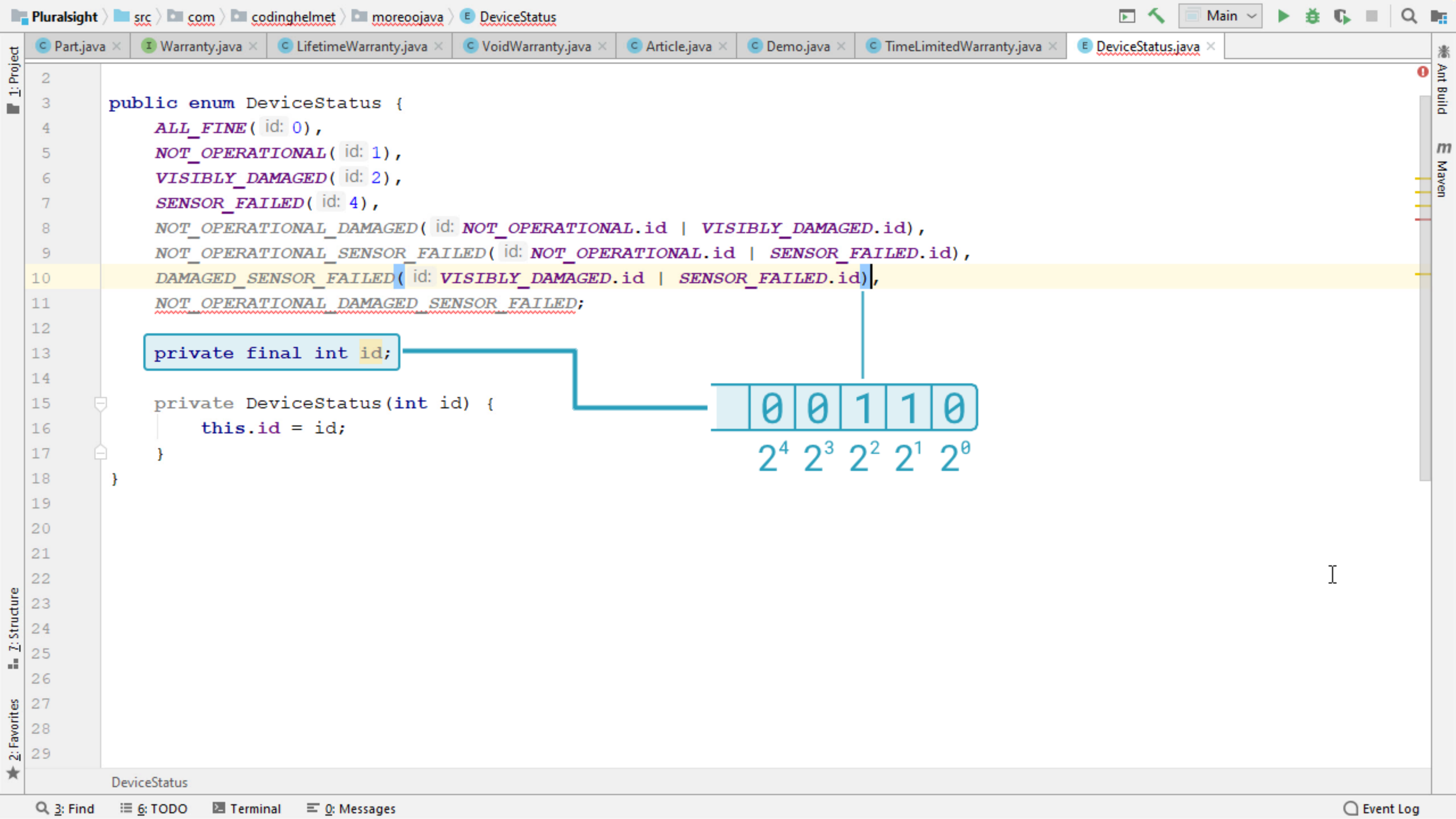


```

public enum DeviceStatus {
    ALL_FINE( id: 0),
    NOT_OPERATIONAL( id: 1),
    VISIBLY_DAMAGED( id: 2),
    SENSOR_FAILED( id: 4),
    NOT_OPERATIONAL_DAMAGED( id: NOT_OPERATIONAL.id | VISIBLY_DAMAGED.id),
    NOT_OPERATIONAL_SENSOR_FAILED,
    DAMAGED_SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED_SENSOR_FAILED;

    private final int id;

    private DeviceStatus(int id) {
        this.id = id;
    }
}
```

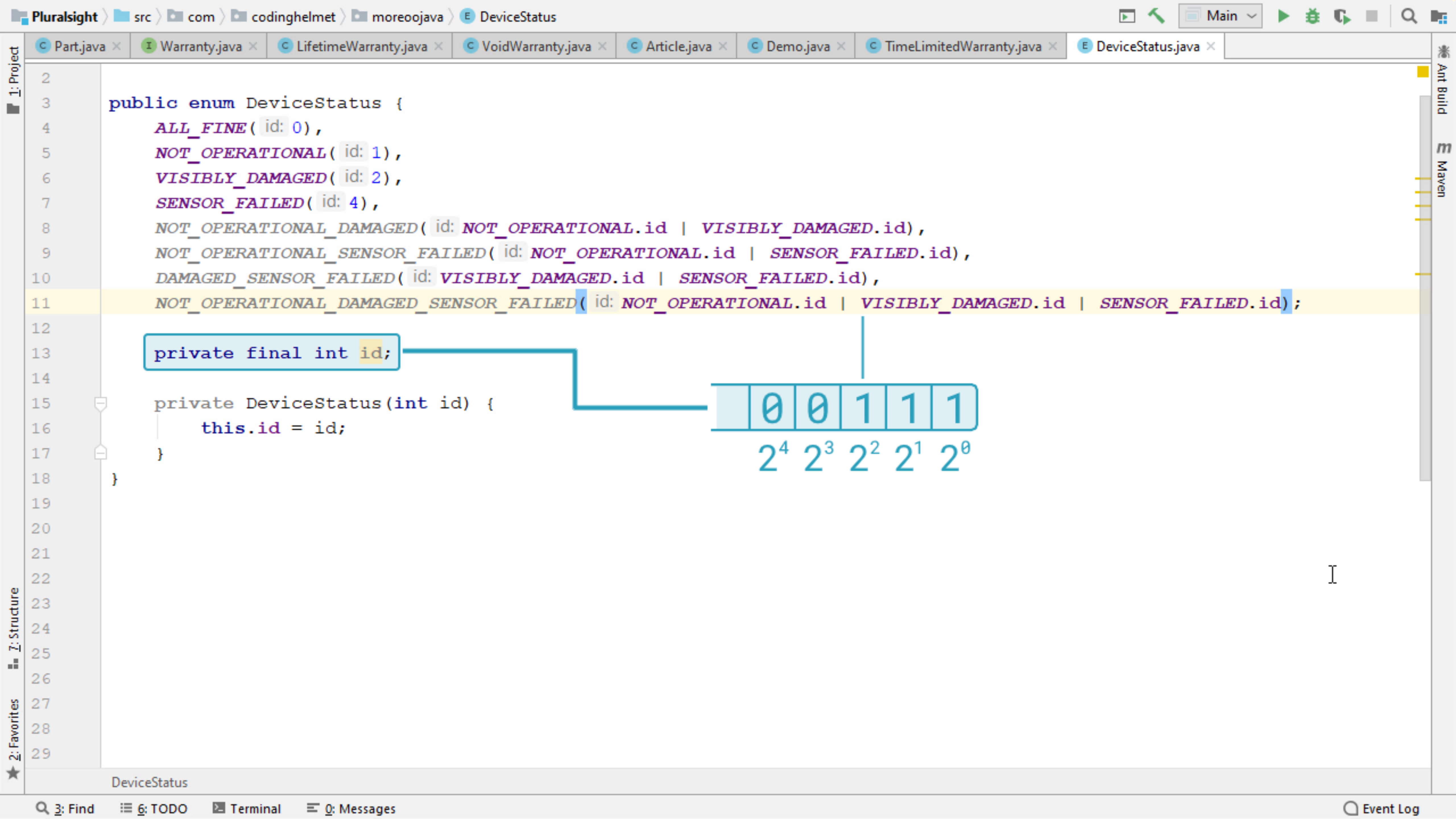
	0	0	0	1	1
	2^4	2^3	2^2	2^1	2^0



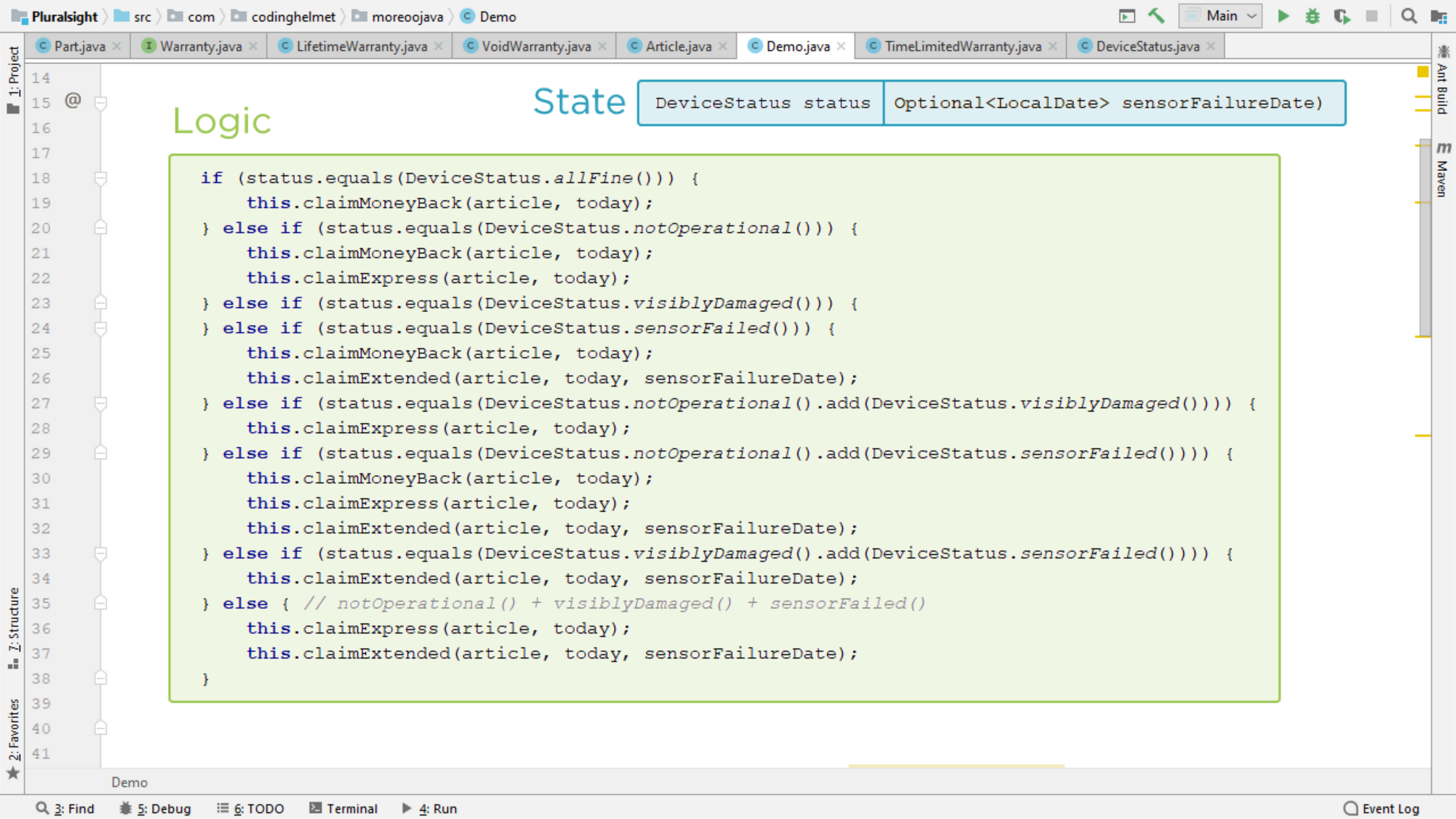
```java

public enum DeviceStatus {
    ALL_FINE( id: 0),
    NOT_OPERATIONAL( id: 1),
    VISIBLY_DAMAGED( id: 2),
    SENSOR_FAILED( id: 4),
    NOT_OPERATIONAL_DAMAGED( id: NOT_OPERATIONAL.id | VISIBLY_DAMAGED.id),
    NOT_OPERATIONAL_SENSOR_FAILED( id: NOT_OPERATIONAL.id | SENSOR_FAILED.id),
    DAMAGED_SENSOR_FAILED,
    NOT_OPERATIONAL_DAMAGED_SENSOR_FAILED;

    private final int id;

    private DeviceStatus(int id) {
        this.id = id;
    }
}
```



```java

public enum DeviceStatus {
    ALL_FINE( id: 0),
    NOT_OPERATIONAL( id: 1),
    VISIBLY_DAMAGED( id: 2),
    SENSOR_FAILED( id: 4),
    NOT_OPERATIONAL_DAMAGED( id: NOT_OPERATIONAL.id | VISIBLY_DAMAGED.id),
    NOT_OPERATIONAL_SENSOR_FAILED( id: NOT_OPERATIONAL.id | SENSOR_FAILED.id),
    DAMAGED_SENSOR_FAILED( id: VISIBLY_DAMAGED.id | SENSOR_FAILED.id),
    NOT_OPERATIONAL_DAMAGED_SENSOR_FAILED;

    private final int id;

    private DeviceStatus(int id) {
        this.id = id;
    }
}
```



```java

public enum DeviceStatus {
    ALL_FINE( id: 0),
    NOT_OPERATIONAL( id: 1),
    VISIBLY_DAMAGED( id: 2),
    SENSOR_FAILED( id: 4),
    NOT_OPERATIONAL_DAMAGED( id: NOT_OPERATIONAL.id | VISIBLY_DAMAGED.id),
    NOT_OPERATIONAL_SENSOR_FAILED( id: NOT_OPERATIONAL.id | SENSOR_FAILED.id),
    DAMAGED_SENSOR_FAILED( id: VISIBLY_DAMAGED.id | SENSOR_FAILED.id),
    NOT_OPERATIONAL_DAMAGED_SENSOR_FAILED( id: NOT_OPERATIONAL.id | VISIBLY_DAMAGED.id | SENSOR_FAILED.id);

    private final int id;

    private DeviceStatus(int id) {
        this.id = id;
    }
}
```

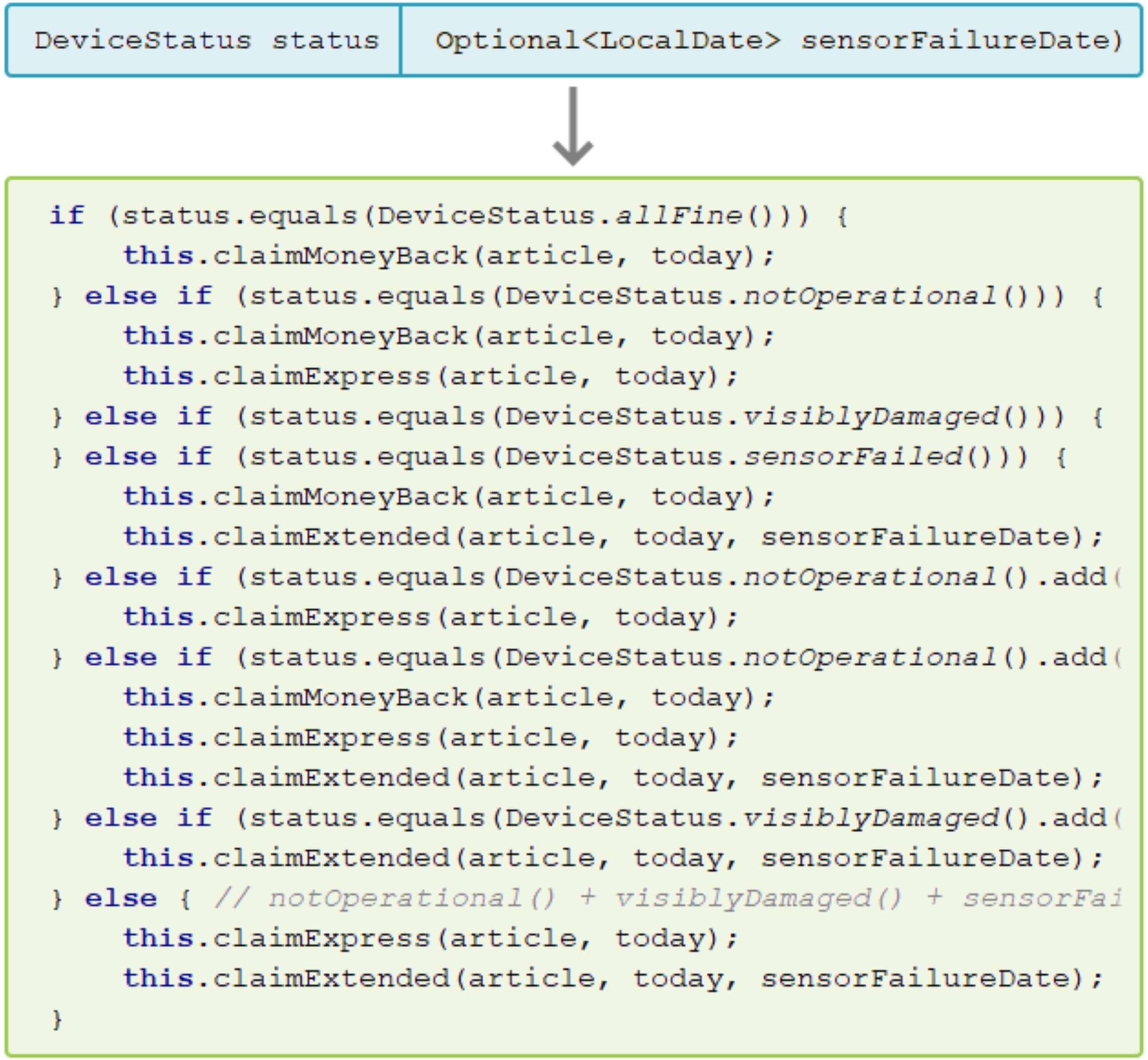


State

DeviceStatus status	Optional<LocalDate> sensorFailureDate)

Logic

```java
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(DeviceStatus.visiblyDamaged()))) {
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(DeviceStatus.sensorFailed()))) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(DeviceStatus.sensorFailed()))) {
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFailed()
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```

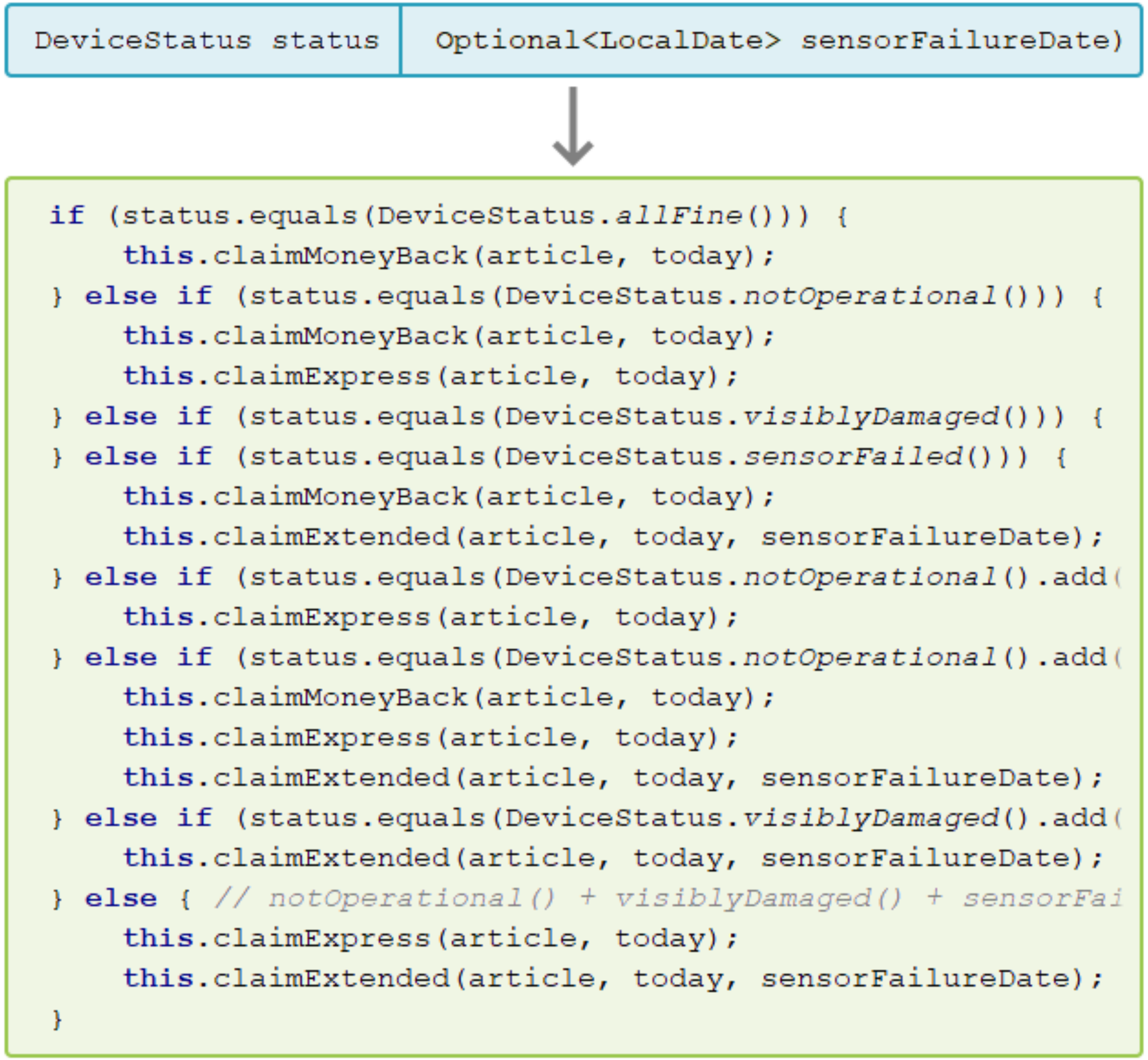


```
DeviceStatus status    Optional<LocalDate> sensorFailureDate)
```

```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```

Poor change handling symptoms

- Implementation must change
- All duplicates must change
- Even on unrelated changes

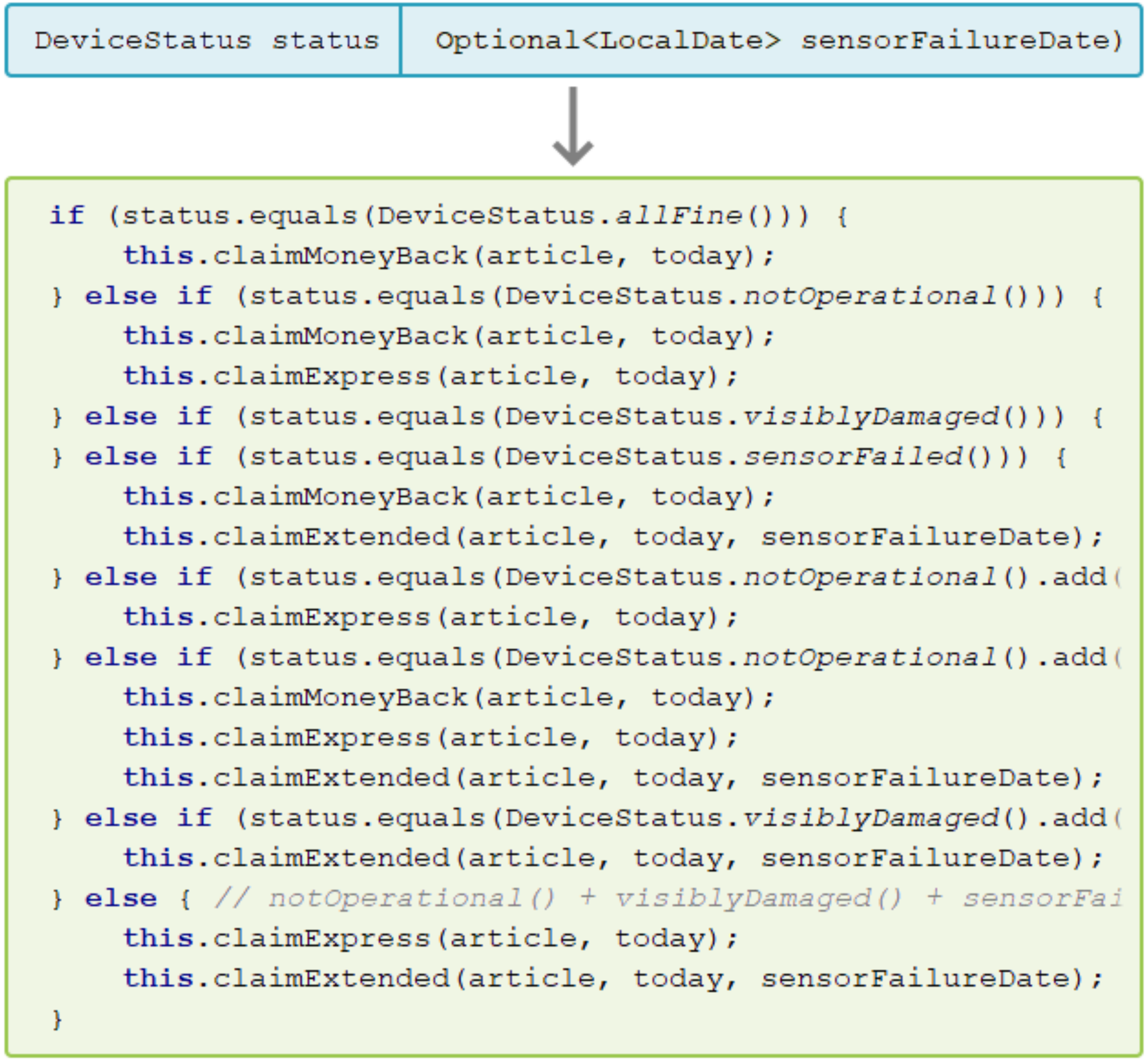


```
DeviceStatus status  Optional<LocalDate> sensorFailureDate)
```

```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```

Visible signs of lacking objects

- Raw state representation
- Consumer implementation
- Hard-coded branching

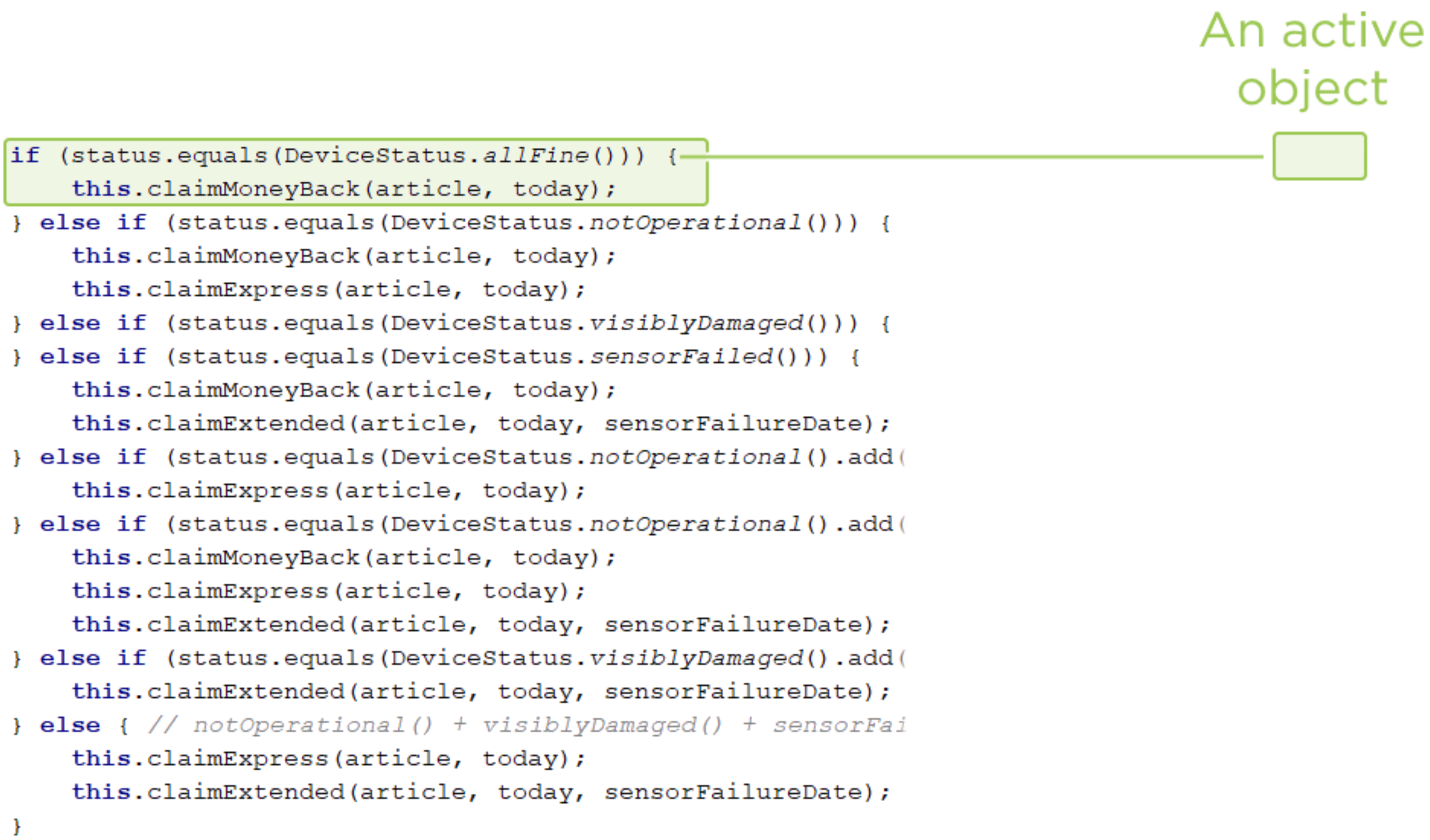


```
DeviceStatus status | Optional<LocalDate> sensorFailureDate)
```

```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```

Expected code qualities

- Flexible behavior
- Support for changing rules



An active object

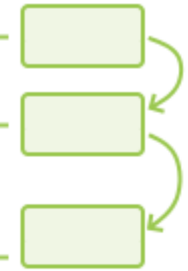
```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```

An active object

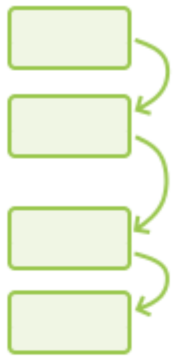
```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```

An active object

```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```



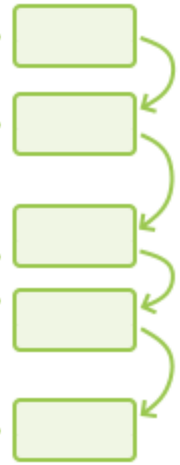
An active object



```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```

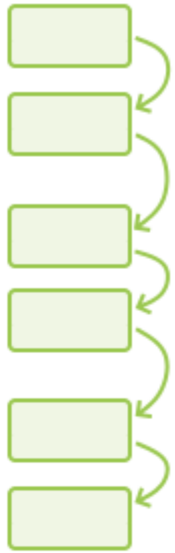
An active object

```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```



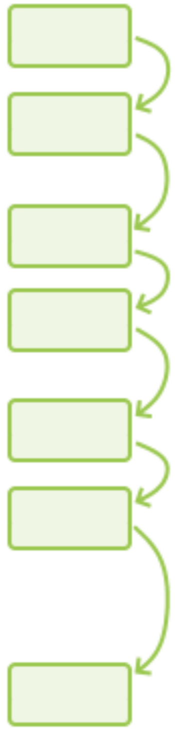
An active object

```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```



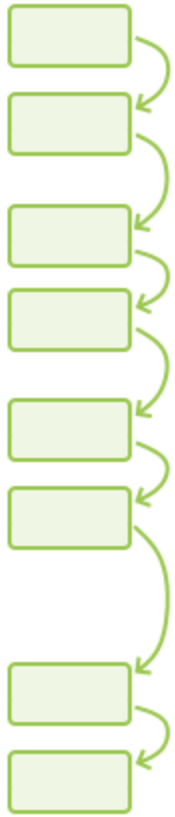
An active object

```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```

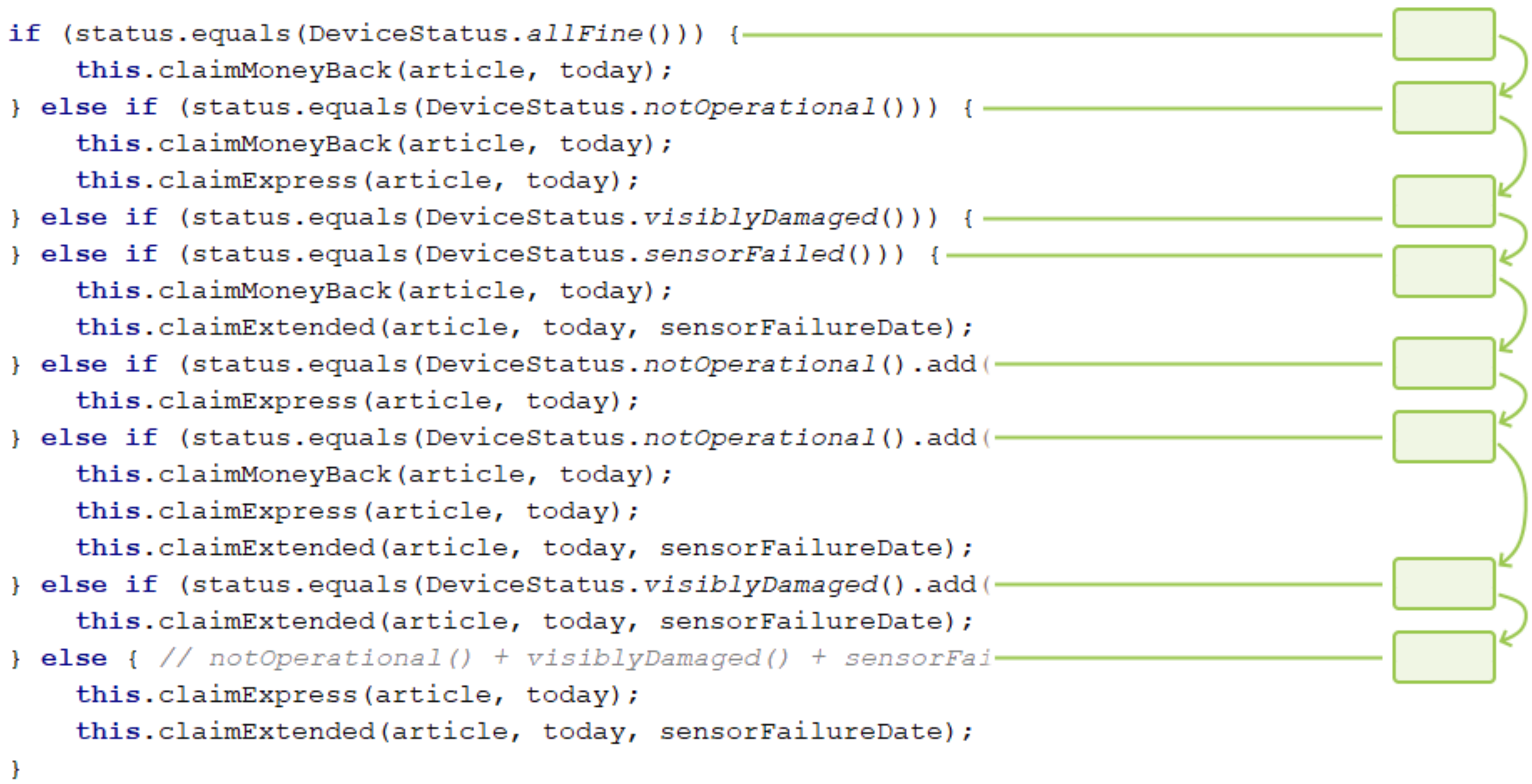


An active object

```java
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```



An active object



```
if (status.equals(DeviceStatus.allFine())) {
    this.claimMoneyBack(article, today);
} else if (status.equals(DeviceStatus.notOperational())) {
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.visiblyDamaged())) {
} else if (status.equals(DeviceStatus.sensorFailed())) {
    this.claimMoneyBack(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimExpress(article, today);
} else if (status.equals(DeviceStatus.notOperational().add(
    this.claimMoneyBack(article, today);
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
} else if (status.equals(DeviceStatus.visiblyDamaged().add(
    this.claimExtended(article, today, sensorFailureDate);
} else { // notOperational() + visiblyDamaged() + sensorFai
    this.claimExpress(article, today);
    this.claimExtended(article, today, sensorFailureDate);
}
```

Chain of rules

Summary

Dealing with non-binary decisions

- Traditional design relies on an `enum`
- Branching implemented with `switch`
- Alternatively, a chain of `if-else` blocks

Summary

Downsides of multiway branching

- What if flags had to be combined?
- That leads to combinatorial explosion
- What if more state should be added?
- That complicates branching conditions
- What if more flag values are added?
- All consuming code must be fixed

Summary

Modeling state with objects

- New state means to add a new class
- Existing classes remain

Summary

Migrating an `enum` to a class

- `enum` values become concrete instances
- Factory functions combine instances
- Objects could still be compared
- Consuming code could remain
- Then let the objects expose behavior
- Join the test pattern with the action

Summary

Next module:
Turning Chained Branching into the Chain of Rule Objects