PEMROGRAMAN LANJUT

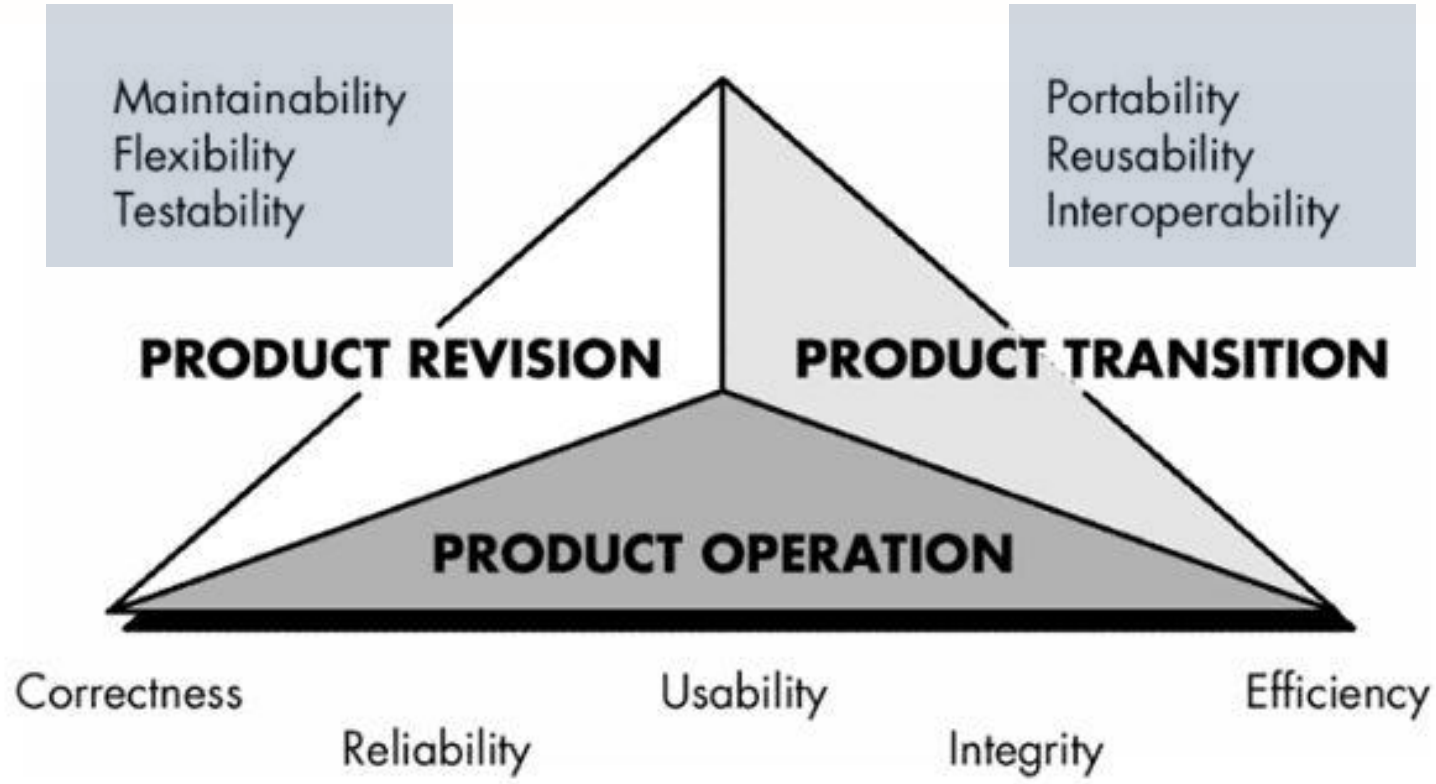
Programming Principles

Oleh Politeknik Elektronika Negeri Surabaya

2021

Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Review



Mc Call Software Quality Metric

Code Conventions

Set of **guidelines** for a specific programming language that **recommend programming style, practices, and methods** for **each aspect** of a program written in that language.

Java Code Convention: Indentation

- Maximum 80-character lines (or 100-character lines according to Google's Android style guide)

Java Code Convention: Indentation

```
function(longExpression1, longExpression2, longExpression3,
         longExpression4, longExpression5);

var = function1(longExpression1,
                function2(longExpression2,
                          longExpression3));
```

Break after comma

Java Code Convention: Indentation

```
function(longExpression1, longExpression2, longExpression3,
         longExpression4, longExpression5);

var = function1(longExpression1,
                function2(longExpression2,
                          longExpression3));
```

Align the new line with the beginning of the expression at the same level on the previous line

Java Code Convention: Indentation

```
longName1 = longName2 * (longName3 + longName4 - longName5)
            + 4 * longname6;

longName1 = longName2 * (longName3 + longName4
                         - longName5) + 4 * longname6;
```

Break before an operator

Java Code Convention: Indentation

```
//CONVENTIONAL INDENTATION
someMethod(int anArg, Object anotherArg, String yetAnotherArg,
           Object andStillAnother) {
    ...
}
```

```
//INDENT 8 SPACES TO AVOID VERY DEEP INDENTS
private static synchronized horkingLongMethodName(int anArg,
        Object anotherArg, String yetAnotherArg,
        Object andStillAnother) {
    ...
}
```

To avoid a very deep indent, you can use 8 spaces to indent the new line as presented in the second example

Java Code Convention: Declaration

- One declaration per line is recommended.
- Put declarations only at the beginning of innermost block that encloses all uses of the variable.

```
void myMethod() {
    int int1 = 0;          // beginning of method block

    if (condition) {
        int int2 = 0;      // beginning of "if" block
        ...
    }
}
```

Java Code Convention: Declaration

```
class Sample extends Object {
    int ivar1;
    int ivar2;

    Sample(int i, int j) {
        ivar1 = i;
        ivar2 = j;
    }

    int emptyMethod() {}

    ...
}
```

Methods are separated by a blank line

Java Code Convention: Statements

- Simple statement: Each line should contain at most one statement
- Compound statement: Lists of statements enclosed in braces

The enclosed statements should be indented one more level than the compound statement

Braces are used around all statements

Java Code Convention: Statements

- A return statement with a value should not use parentheses unless they make the return value more obvious in some way.

```
return;

return myDisk.size();

return (size ? size : defaultSize);
```

Java Code Convention: Naming Convention

Item	Capitalization	Example

Package	All-lowercase ASCII letters.	`com.sun.eng`
		`com.apple.quicktime.v2`
		`edu.cmu.cs.bovik.cheese`

Class	Mixed case with the first letter of each internal word capitalized ➔ Pascal Case It must use nouns.	`class Raster;` `class ImageSprite;`
Interface	Mixed case with the first letter of each internal word capitalized ➔ Pascal Case	`interface RasterDelegate;` `interface Storing;`

Method	Mixed case with the first letter lowercase, with the first letter of each internal word capitalized ➔ Camel Case
	It must use verbs.
		`run();`
		`runFast();`
		`getBackground();`

Java Code Convention: Naming Convention

Item	Capitalization	Example
Variable	Mixed case with a lowercase first letter. Internal words start with capital letters. It should not start with underscore _ or dollar sign $ characters, even though both are allowed.	`int playerScore;` `String name;`

Constant	All uppercase with words separated by underscores ("_") ➔ Snake Case.	`static final int MIN_WIDTH = 4;`
		`static final int MAX_WIDTH = 999;`
		`static final int GET_THE_CPU = 1;`

Java Code Convention: Exceptions

```
try {
    someComplicatedIOFunction();          // may throw IOException
    someComplicatedParsingFunction();     // may throw ParsingException
    someComplicatedSecurityFunction();    // may throw SecurityException
    // phew, made it all the way
} catch (Exception e) {                   // I'll just catch all exceptions
    handleError();                        // with one generic handler!
}
```

```
void setServerPort(String value) {
    try {
        serverPort = Integer.parseInt(value);
    } catch (NumberFormatException e) { }
}
```

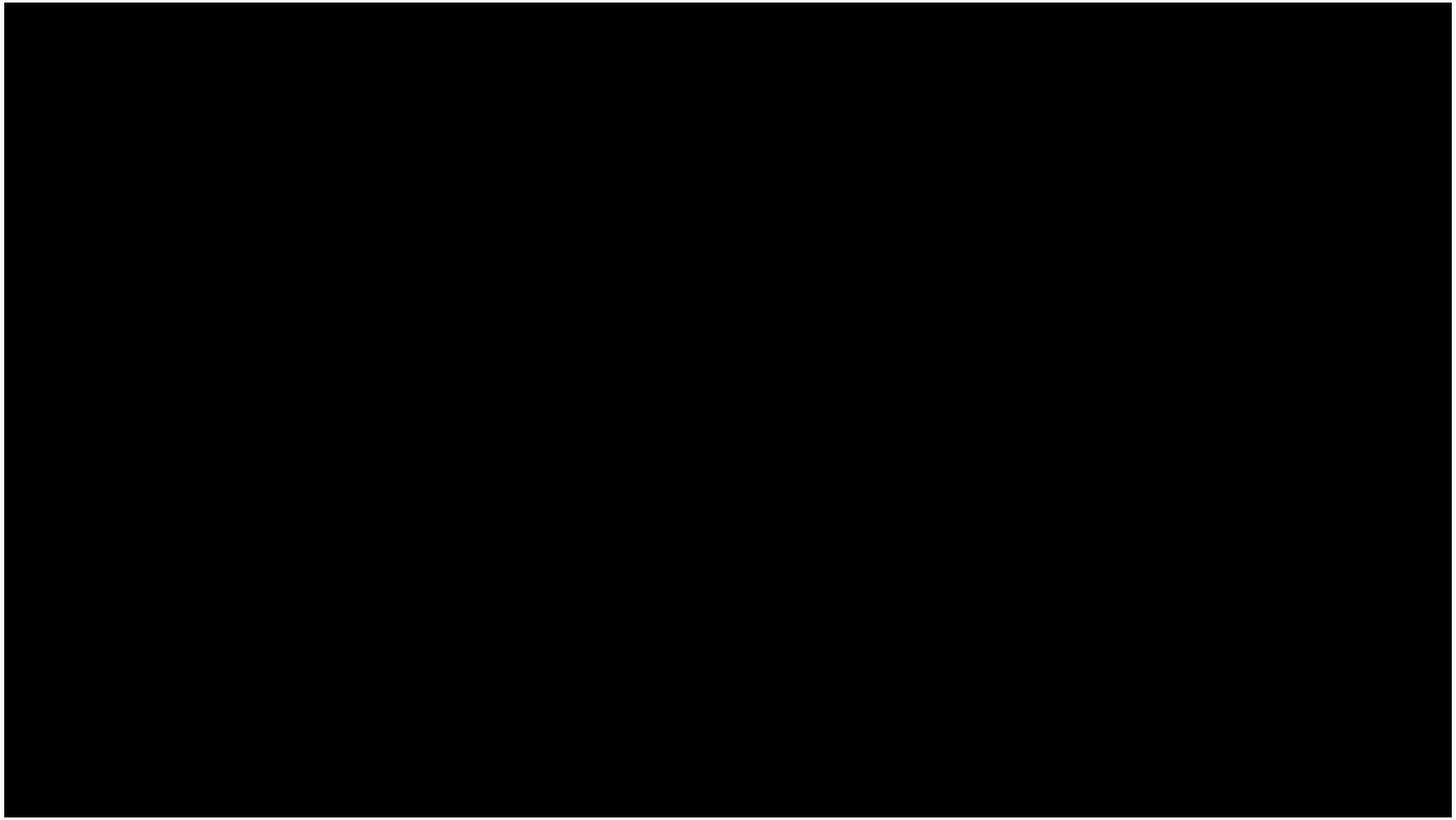
Java Code Convention: Don't Use Wildcard Imports

```
import foo.*;
```

```
import foo.Bar;
```

End of Review

DRY (Don’t Repeat Yourself)



DRY (Don't Repeat Yourself)

- Find and eliminate duplication wherever you can.

```
public void method1() {
    System.out.println("Saya bisa Clean Code. Saya yakin. InsyaAllah..");
    System.out.println("Saya bisa Clean Code. Saya yakin. InsyaAllah..");
    System.out.println("Saya bisa Clean Code. Saya yakin. InsyaAllah..");
    System.out.println("Saya bisa Clean Code. Saya yakin. InsyaAllah..");
    System.out.println("Saya bisa Clean Code. Saya yakin. InsyaAllah..");
    System.out.println("Saya bisa Clean Code. Saya yakin. InsyaAllah..");
    System.out.println("Saya bisa Clean Code. Saya yakin. InsyaAllah..");
}
```

Smells:
- Duplicate code
- Data clumps

DRY (Don't Repeat Yourself)

- Find and eliminate duplication wherever you can.

```
public void method2() {
    System.out.println("Saya anak ke-1.");
    System.out.println("Saya anak ke-2.");
    System.out.println("Saya anak ke-3.");
    System.out.println("Saya anak ke-4.");
    System.out.println("Saya anak ke-5.");
}
```

```
public void method3() {
    System.out.println("Keturunan ke-1 disebut anak.");
    System.out.println("Keturunan ke-2 disebut cucu.");
    System.out.println("Keturunan ke-3 disebut cicit.");
    System.out.println("Keturunan ke-4 disebut canggah.");
    System.out.println("Keturunan ke-5 disebut anggas.");
}
```

DRY (Don't Repeat Yourself)

- The most obvious form of duplication is when you have clumps of identical code in various places.

Other example:
https://pastebin.com/ikvpT7pX

```
public void method4() {
    People people1 = new People();
    people1.setName("Ariana");
    people1.setAge(18);
    people1.setHeight(178);
    people1.printInfo();

    People people2 = new People();
    people2.setName("Baharudin");
    people2.setAge(27);
    people2.setHeight(180);
    people2.printInfo();

    People people3 = new People();
    people3.setName("Cintya");
    people3.setAge(10);
    people3.setHeight(160);
    people3.printInfo();
}
```

DRY (Don't Repeat Yourself)

- A more subtle form is the switch/case or if/else chain that appears again and again in various modules, always testing for the same set of conditions. These should be replaced with polymorphism.

```
public double calculateSalary(String status){
    switch (status) {
        case "intern":
            return 0.8*baseSalary;
        case "manager":
            return baseSalary + lengthofWork* 500000 + bonus;
        case "senior employee":
            return baseSalary + lengthofWork* 500000;
        default:
            return baseSalary;
    }
}
```

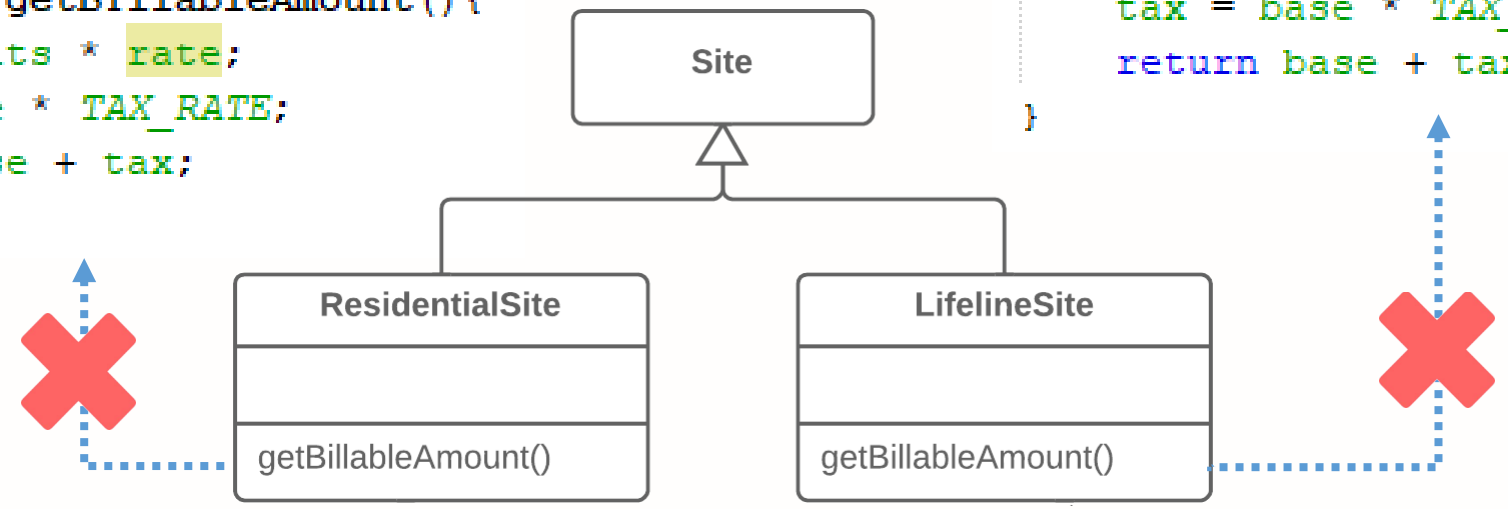
```
public double calculateHoliday(String status){
    switch (status) {
        case "intern":
            return 0;
        case "manager":
            return 24;
        case "senior employee":
            return 18;
        default:
            return 12;
    }
}
```

DRY (Don't Repeat Yourself)

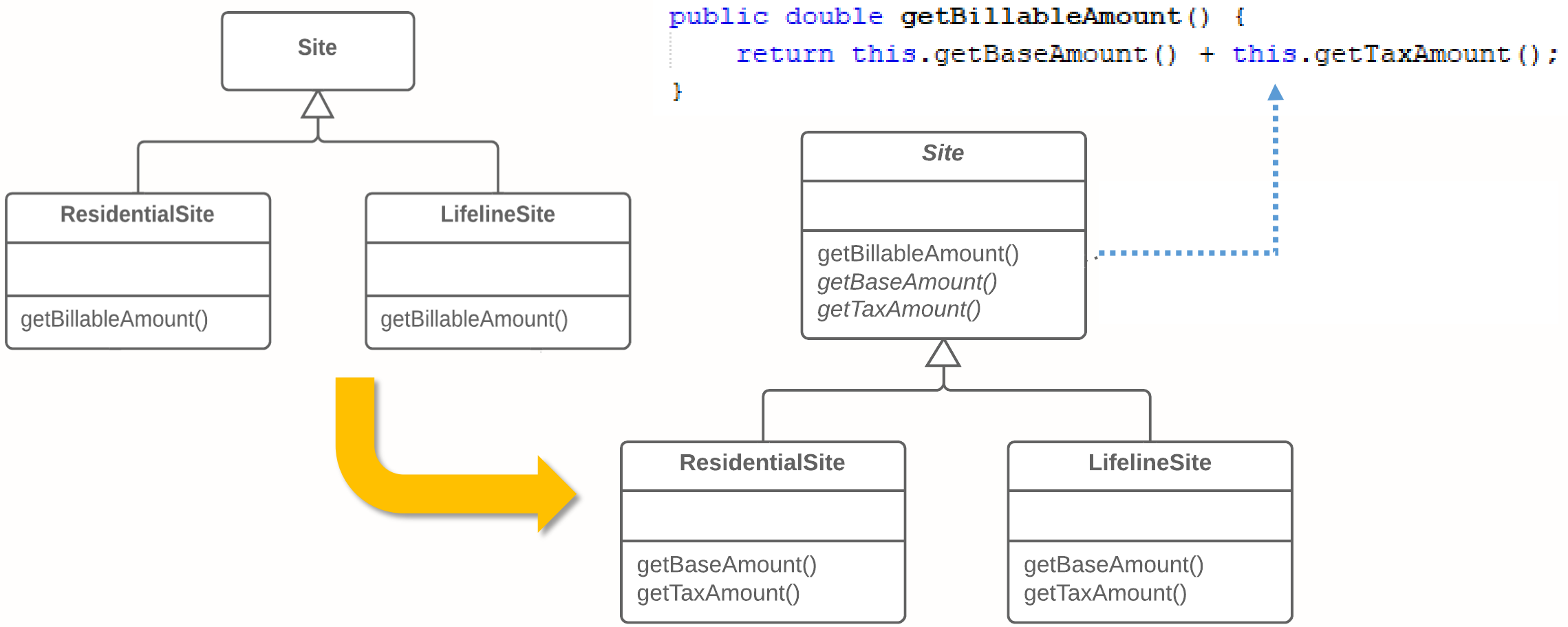
- Still more subtle are the modules that have similar algorithms, but that don't share similar lines of code. This is still duplication.

```
public double getBillableAmount(){
    base = units * rate;
    tax = base * TAX_RATE;
    return base + tax;
}
```

```
public double getBillableAmount() {
    base = units * rate * 0.5;
    tax = base * TAX_RATE * 0.2;
    return base + tax;
}
```



DRY (Don't Repeat Yourself)



KISS (Keep It Simple St**id)

- Keep the code simple and clear, making it easy to understand.

```
public void toogleExample(boolean a) {
    boolean b;
    if (a == false) {
        b = true;
    } else {
        b = false;
    }
    System.out.println(b);
}
```

```
public boolean isSucceed(int point) {
    if (point >= 7) {
        return true;
    } else {
        return false;
    }
}
```

KISS (Keep It Simple St**id)

- Keep the code simple and clear, making it easy to understand.

```
public double getPayAmount() {
    double result;
    if (isDead()) {
        result = deadAmount();
    } else {
        if (isSeparated()) {
            result = separatedAmount();
        } else {
            if (isRetired()) {
                result = retiredAmount();
            } else {
                result = normalPayAmount();
            }
        }
    }
    return result;
}
```

YAGNI (You Are Not Gonna Need It)

- Remove any parts which are unnecessary.
- Do not implement something until it is needed.

Smells:
- Dead code
- Speculative generality
- Lazy class
- Comments

```
/**
 * @param idShape, options: 2D Shapes: rectangle, square, circle 3D Shapes:
 * cube, cuboid, cone, sphere
 * @param factor1
 * @param factor2
 * @return area for 2D Shape
 */
public double calculateArea(String idShape, double factor1, double factor2) {
    double result = 0;

    switch (idShape) {
        case "rectangle":
            result = factor1 * factor2; //width * height
            break;
        case "square":
            result = factor1 * factor1; //side * side
            break;
        case "circle":
            result = 3.14 * factor1 * factor1; //PI * radius^2
            break;
    }
    return result;
}
```

References

- Rasyid Institute. Modul Workshop Clean Code. 2019.
- Fowler, Martin. Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, 1999.
- Martin, Robert C. Clean Code: A Handbook of Agile Software Craftsmanship. Pearson. 2008.
- https://refactoring.guru/
- http://www.youtube.com/watch?v=leIm-OLXQ7Q

bridge to the future

http://www.eepis-its.edu