PITECH+PLUS

Coding the right way

Best practices learned in industry while coding the wrong way

PITECH+PLUS

Senior software engineer and Clean Code advocate

Software engineer and PhD @ UBB

PITECH PLUS

Contents

PITECH+PLUS

What's more important:
a functional project
or
a clean codebase?

PITECH+PLUS

Case study

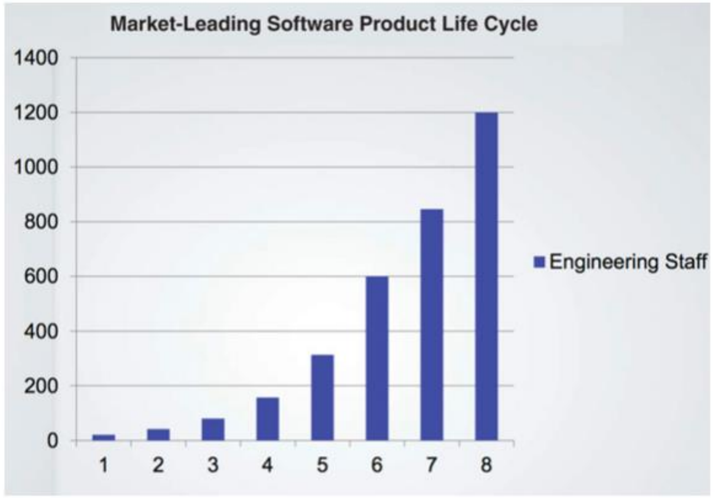


Figure 1.1 Growth of the engineering staff

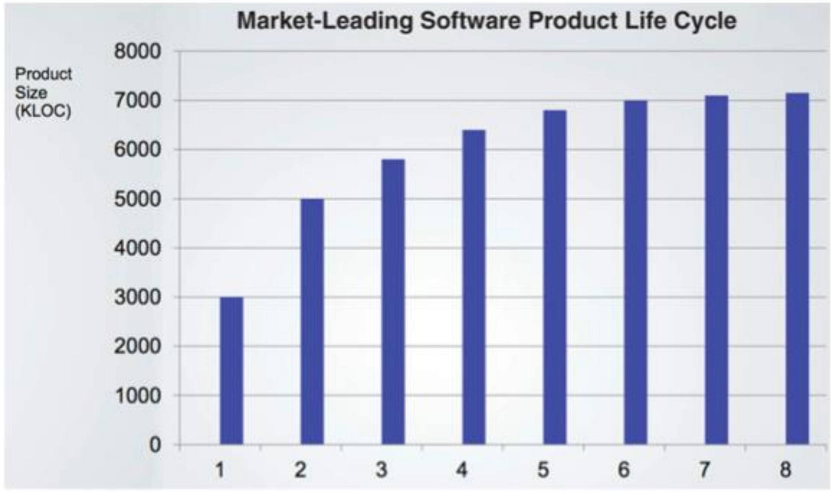


Figure 1.2 Productivity over the same period of time

PITECH+PLUS

Case study

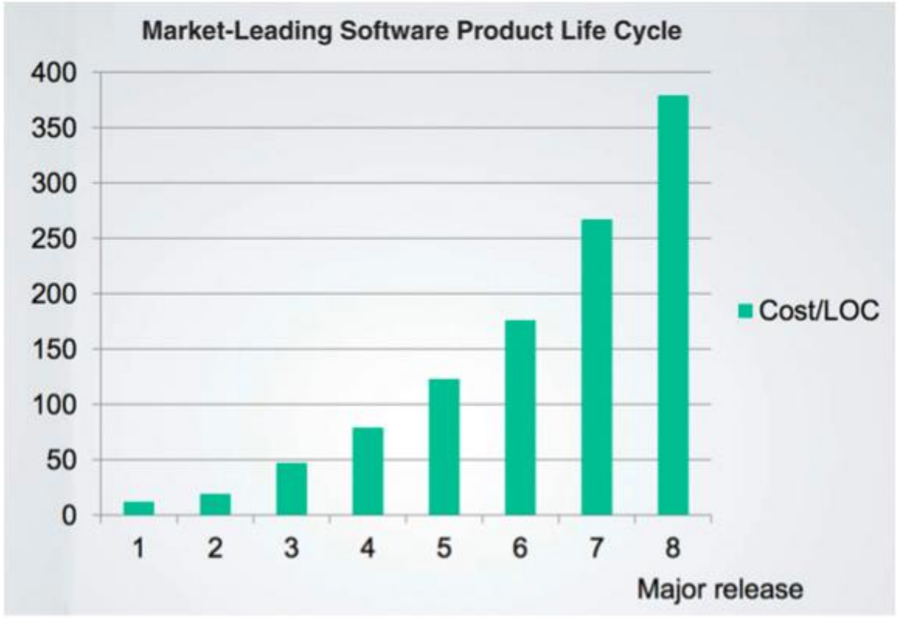


Figure 1.3 Cost per line of code over time

PITECH+PLUS

Grady Booch, author of *Object Oriented Analysis and Design with Applications*

Clean code is simple and direct. Clean code reads like well-written prose. Clean code never obscures the designer's intent but rather is full of crisp abstractions and straightforward lines of control.

Bjarne Stroustrup, inventor of C++ and author of *The C++ Programming Language*

I like my code to be elegant and efficient. The logic should be straightforward to make it hard for bugs to hide, the dependencies minimal to ease maintenance, error handling complete according to an articulated strategy, and performance close to optimal so as not to tempt people to make the code messy with unprincipled optimizations. Clean code does one thing well.

"Big" Dave Thomas, founder of OTI, godfather of the Eclipse strategy

Clean code can be read, and enhanced by a developer other than its original author. It has unit and acceptance tests. It has meaningful names. It provides one way rather than many ways for doing one thing. It has minimal dependencies, which are explicitly defined, and provides a clear and minimal API. Code should be literate since depending on the language, not all necessary information can be expressed clearly in code alone.

PITECH+PLUS

Meaningful names

PITECH+PLUS

Meaningful names

Rules for creating meaningful names:

- ➢ **Use intention-revealing names**
- ➢ **Use searchable names**
- ➢ **Class names should be formed by a noun or noun phrase like Customer, Job, Account, UserProfile etc**
- ➢ **Method names should contain a verb: save(), makePayment(), deleteUser()**
- ➢ **Pick one word per concept / Use the same terms consistently**

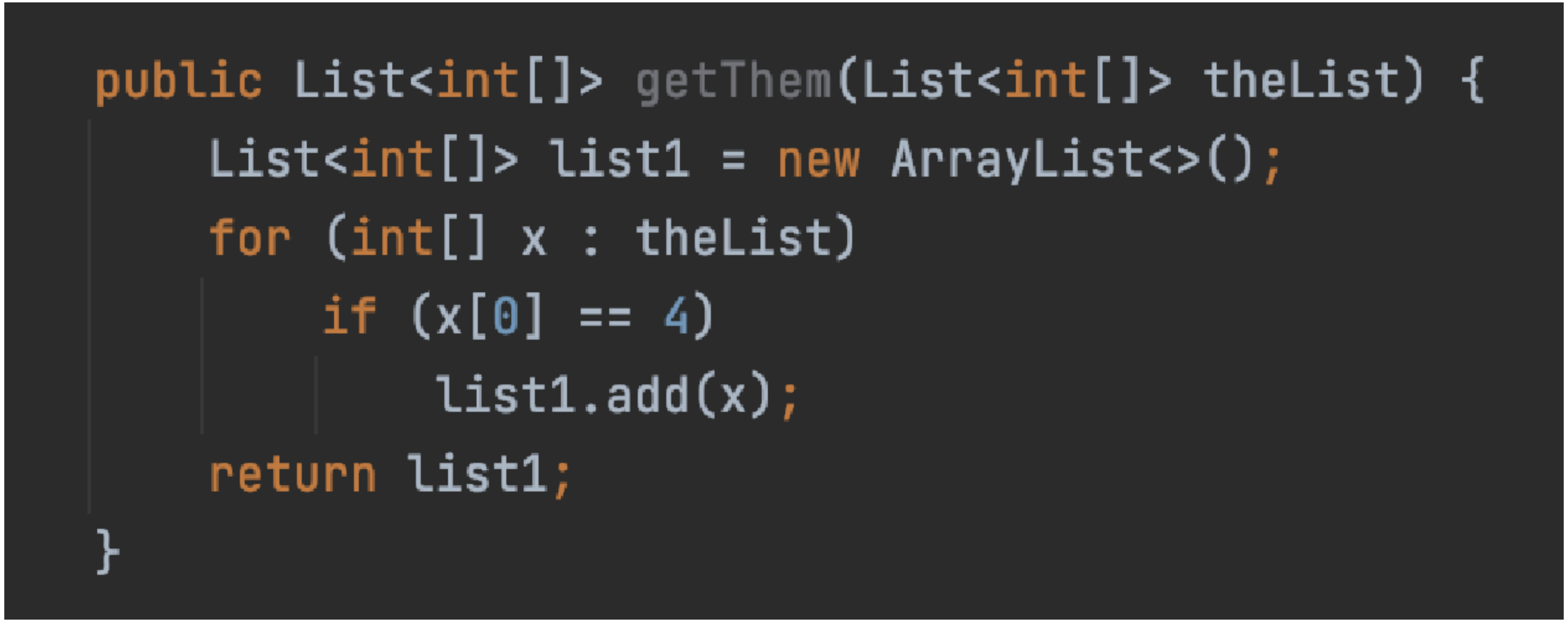
Meaningful names - examples

```
int d;     //elapsed time in days
```

```
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
int fileAgeInDays;
```

PITECH+PLUS

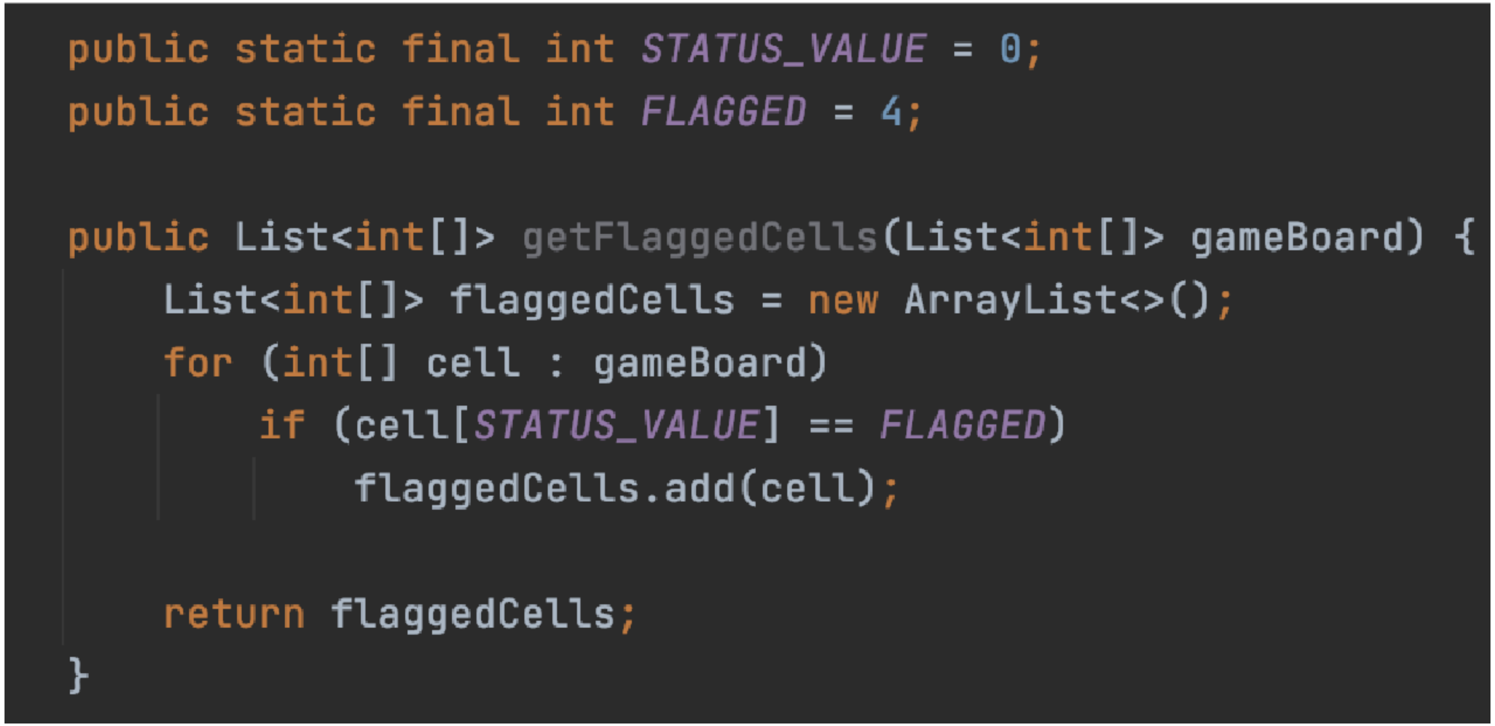
Meaningful names - examples



```java
public List<int[]> getThem(List<int[]> theList) {
    List<int[]> list1 = new ArrayList<>();
    for (int[] x : theList)
        if (x[0] == 4)
            list1.add(x);
    return list1;
}
```

PITECH+PLUS

Meaningful names - examples



```java
public static final int STATUS_VALUE = 0;
public static final int FLAGGED = 4;

public List<int[]> getFlaggedCells(List<int[]> gameBoard) {
    List<int[]> flaggedCells = new ArrayList<>();
    for (int[] cell : gameBoard)
        if (cell[STATUS_VALUE] == FLAGGED)
            flaggedCells.add(cell);

    return flaggedCells;
}
```

PITECH+PLUS

Functions

PITECH+PLUS

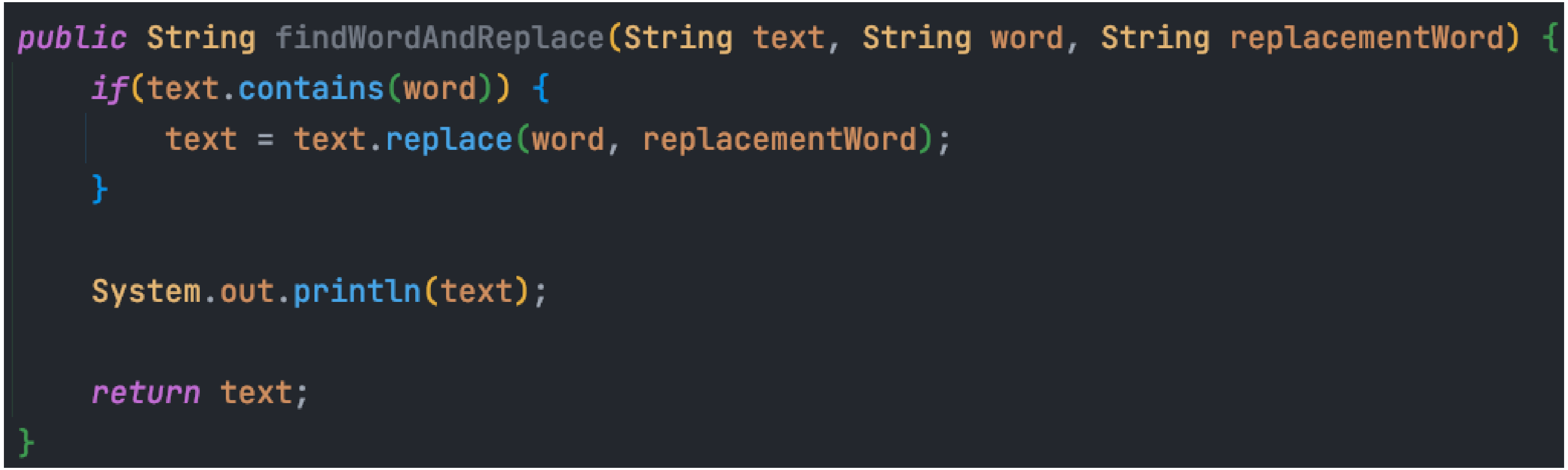
Functions

In a common program structure, the most important rules when writing a function are:

- **Functions should be small**
- **Have one reason to change**
- **One level of abstraction per function**
- **Avoid side effects**
- **Function arguments (not too many)**
- **Prefer exceptions to returning error codes**

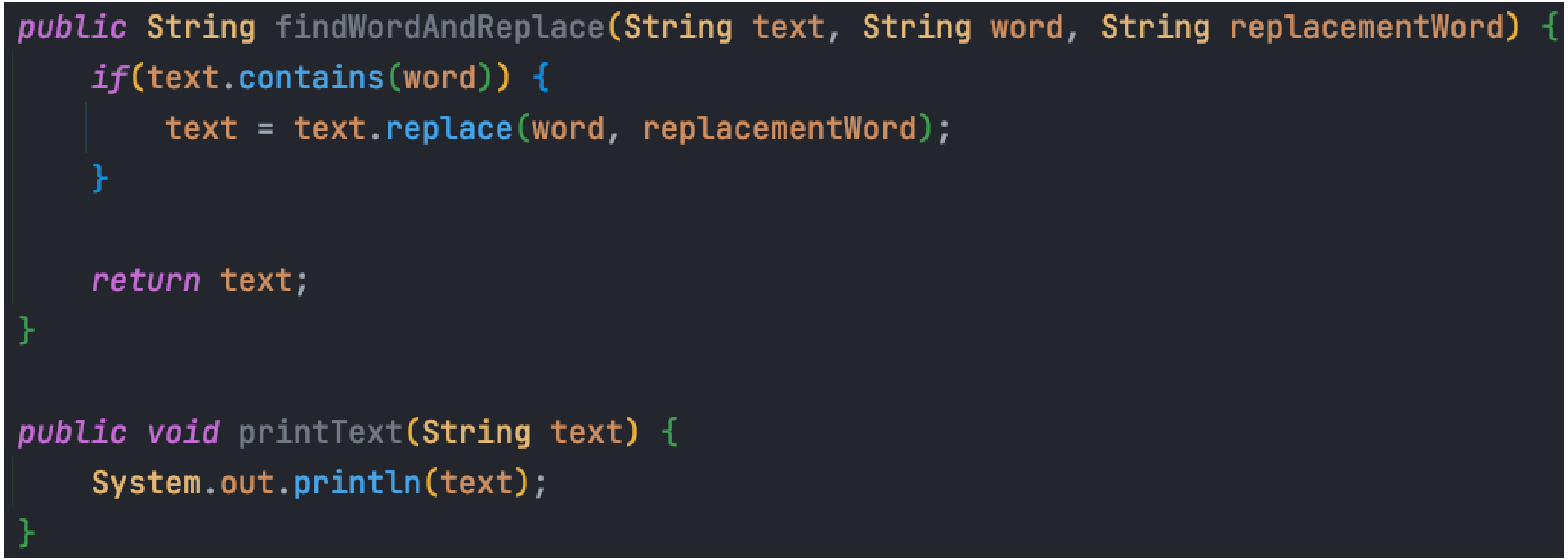
PITECH+PLUS

Functions - examples



```java
public String findWordAndReplace(String text, String word, String replacementWord) {
    if(text.contains(word)) {
        text = text.replace(word, replacementWord);
    }

    System.out.println(text);

    return text;
}
```

Functions - examples



```java
public String findWordAndReplace(String text, String word, String replacementWord) {
    if(text.contains(word)) {
        text = text.replace(word, replacementWord);
    }

    return text;
}

public void printText(String text) {
    System.out.println(text);
}
```

PITECH+PLUS

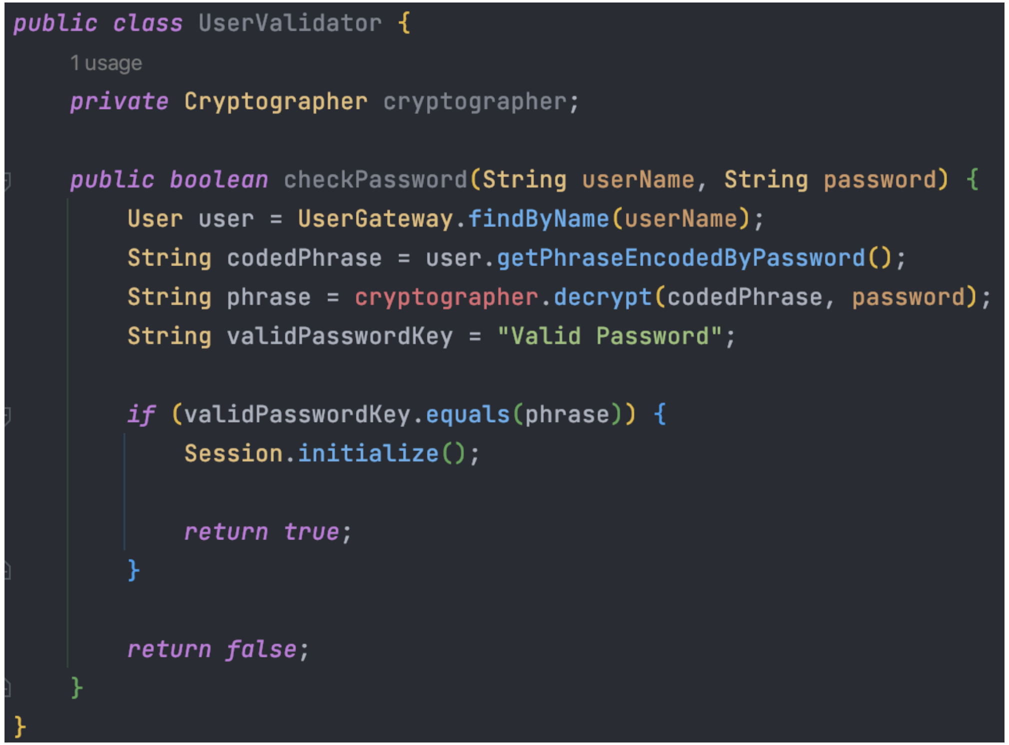
Pure functions

A function is called as pure if it satisfies these two principles:

- ➢**The return value of the function depends only on the input parameters passed to the function.**
- ➢**The execution of the function has no side effects.**

PITECH+PLUS

Functions - examples



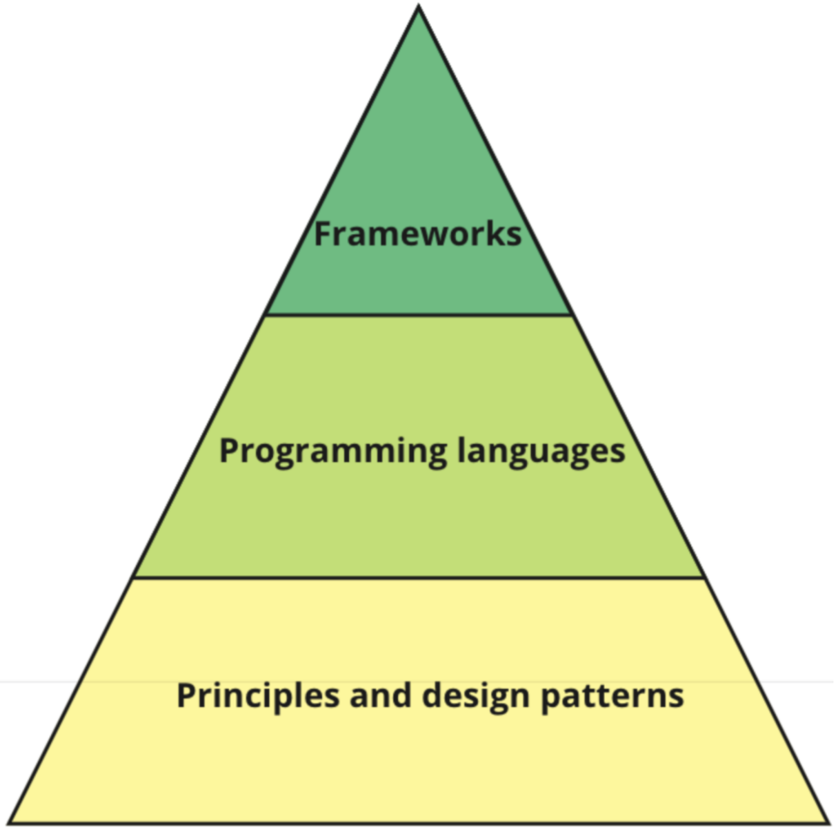
```
public class UserValidator {
    1 usage
    private Cryptographer cryptographer;

    public boolean checkPassword(String userName, String password) {
        User user = UserGateway.findByName(userName);
        String codedPhrase = user.getPhraseEncodedByPassword();
        String phrase = cryptographer.decrypt(codedPhrase, password);
        String validPasswordKey = "Valid Password";

        if (validPasswordKey.equals(phrase)) {
            Session.initialize();

            return true;
        }

        return false;
    }
}
```

PITECH+PLUS

Programming Principles

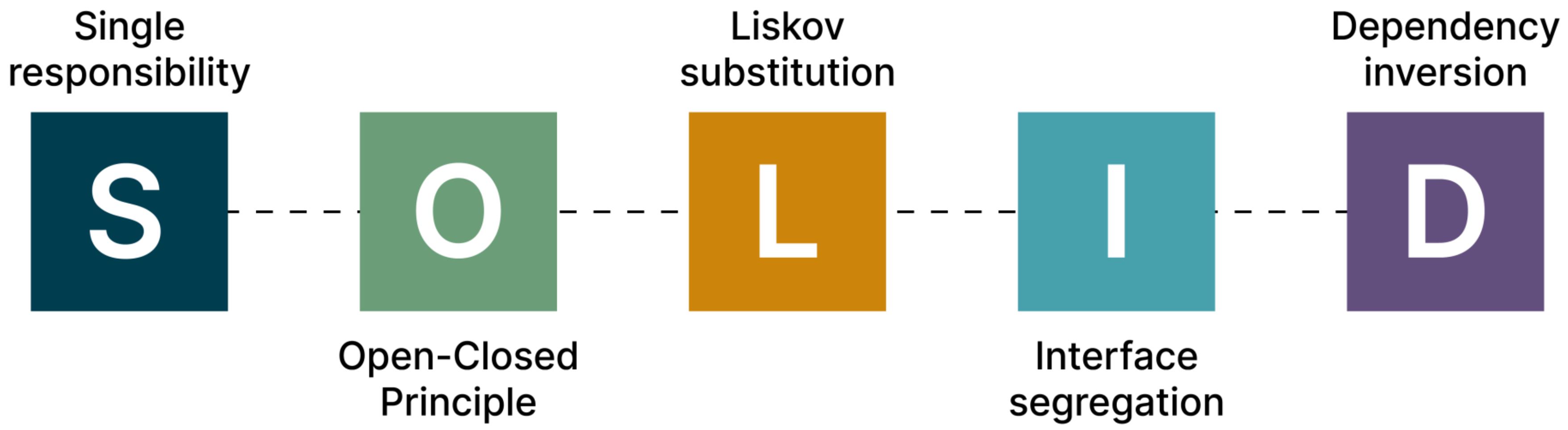
PITECH+PLUS

Why?



PITECH PLUS

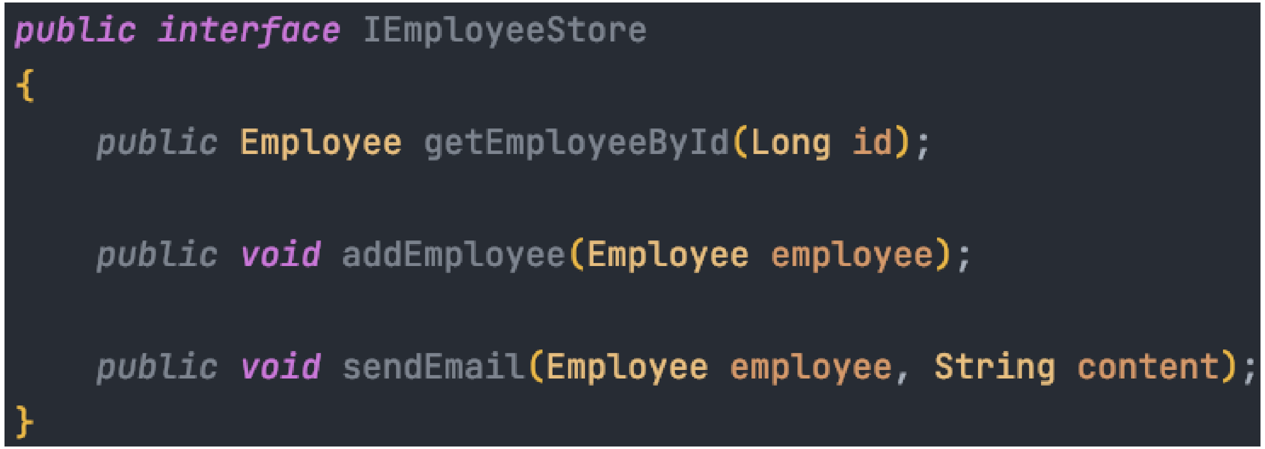
SOLID Principles



PITECH PLUS

Single responsibility principle

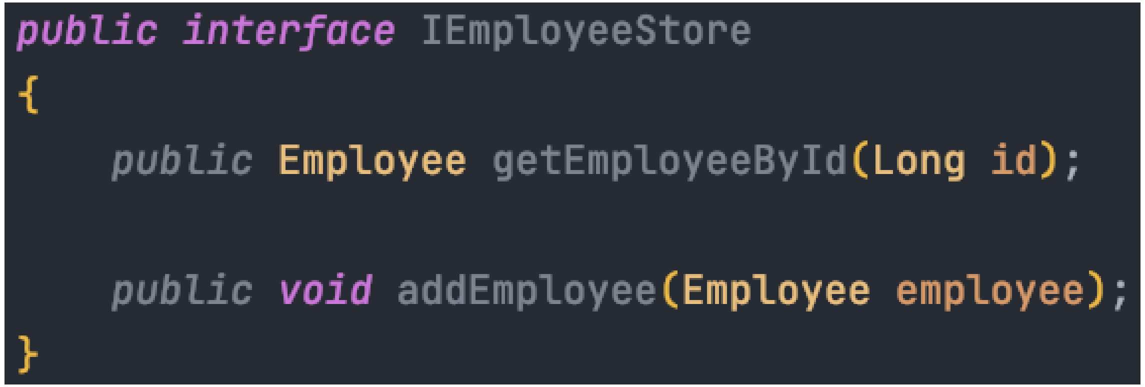
A class should have only one reason to change.



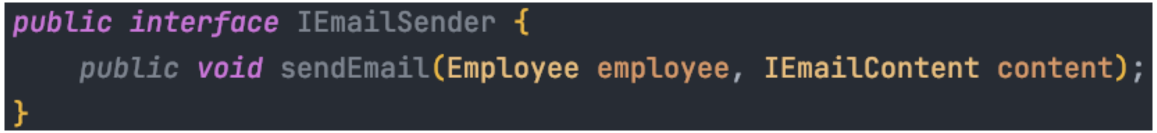
```
public interface IEmployeeStore
{
    public Employee getEmployeeById(Long id);

    public void addEmployee(Employee employee);

    public void sendEmail(Employee employee, String content);
}
```

PITECH PLUS

Single responsibility principle



```
public interface IEmployeeStore
{
    public Employee getEmployeeById(Long id);

    public void addEmployee(Employee employee);
}
```

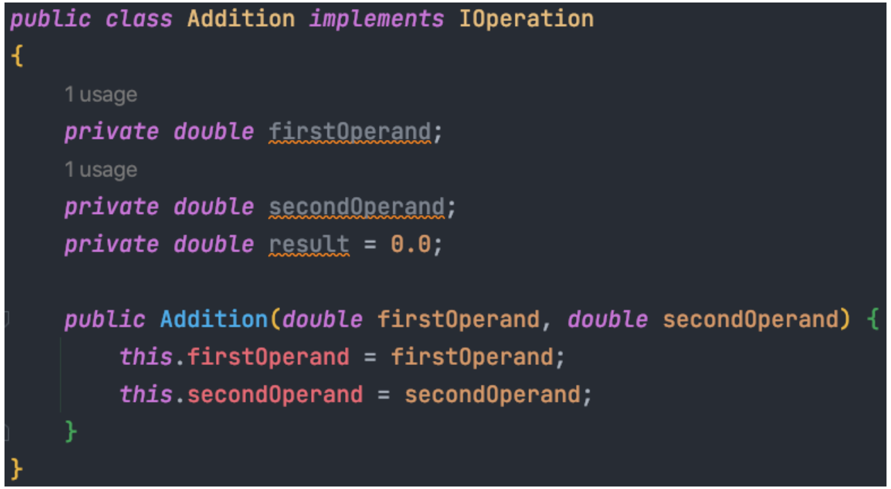


```
public interface IEmailSender {
    public void sendEmail(Employee employee, IEmailContent content);
}
```

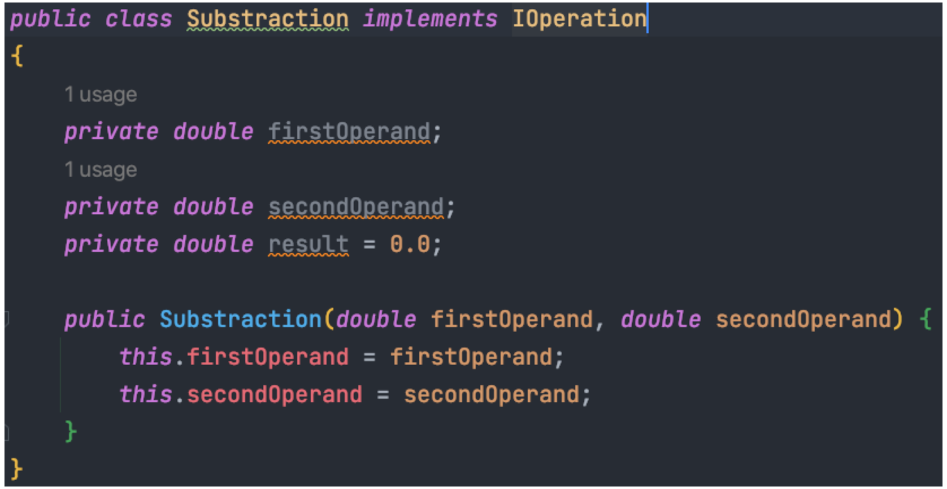
PITECH+PLUS

Open closed principle

A class should be opened for extension, but closed for modification



```
public class Addition implements IOperation
{
    1 usage
    private double firstOperand;
    1 usage
    private double secondOperand;
    private double result = 0.0;

    public Addition(double firstOperand, double secondOperand) {
        this.firstOperand = firstOperand;
        this.secondOperand = secondOperand;
    }
}
```

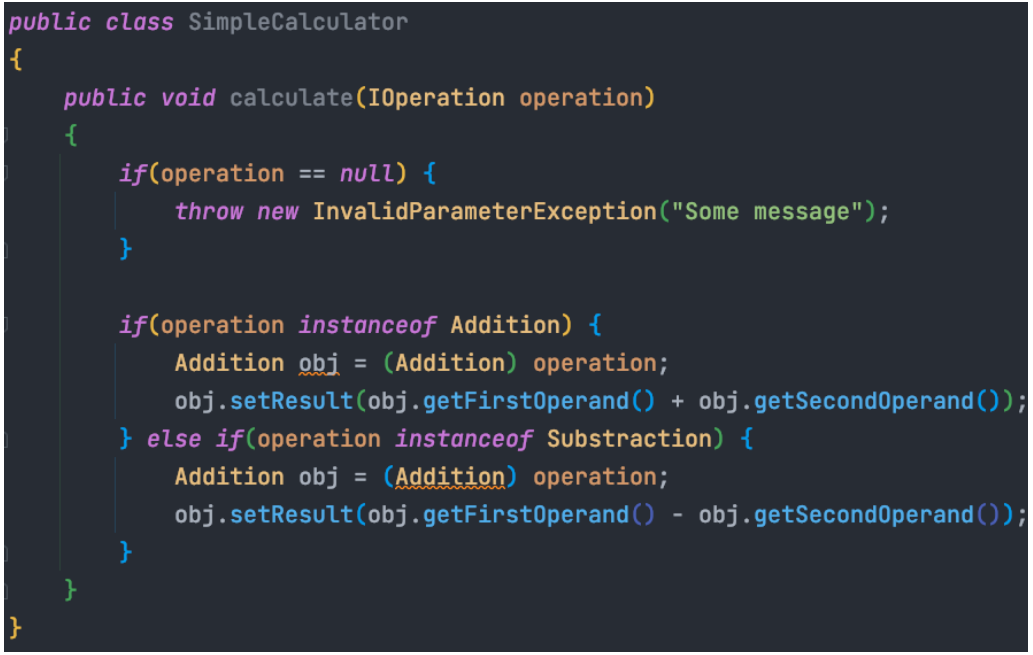


```
public class Substraction implements IOperation
{
    1 usage
    private double firstOperand;
    1 usage
    private double secondOperand;
    private double result = 0.0;

    public Substraction(double firstOperand, double secondOperand) {
        this.firstOperand = firstOperand;
        this.secondOperand = secondOperand;
    }
}
```

PITECH+PLUS

Open closed principle

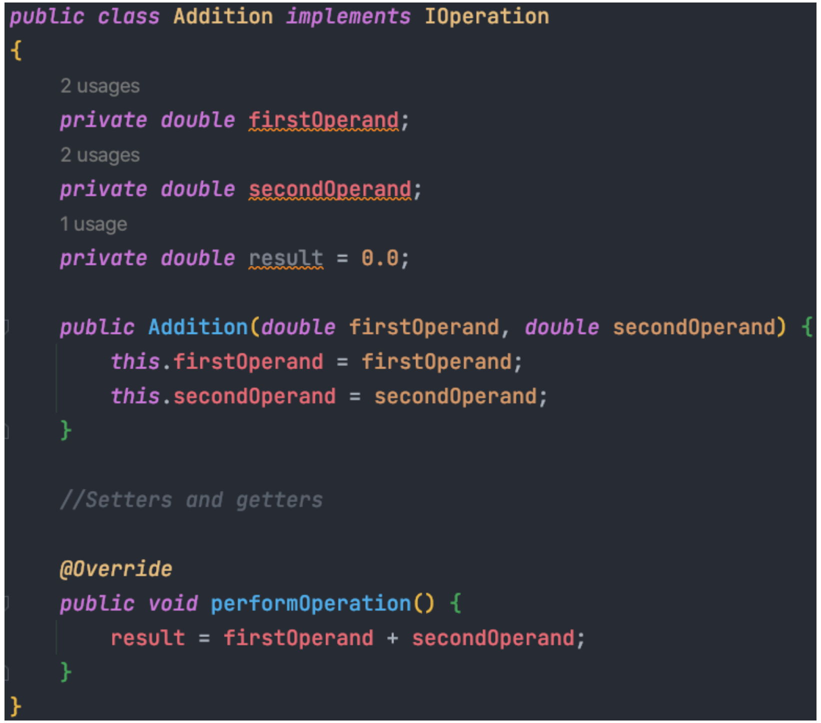
A class should be opened for extension, but closed for modification



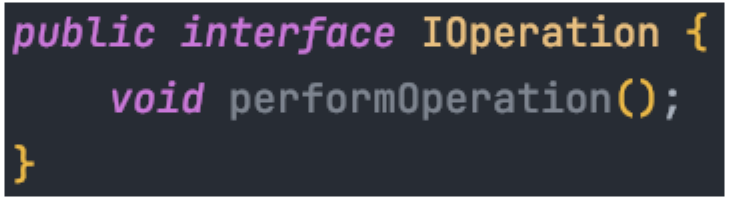
```java
public class SimpleCalculator
{
    public void calculate(IOperation operation)
    {
        if(operation == null) {
            throw new InvalidParameterException("Some message");
        }

        if(operation instanceof Addition) {
            Addition obj = (Addition) operation;
            obj.setResult(obj.getFirstOperand() + obj.getSecondOperand());
        } else if(operation instanceof Substraction) {
            Addition obj = (Addition) operation;
            obj.setResult(obj.getFirstOperand() - obj.getSecondOperand());
        }
    }
}
```

PITECH+PLUS

Open closed principle



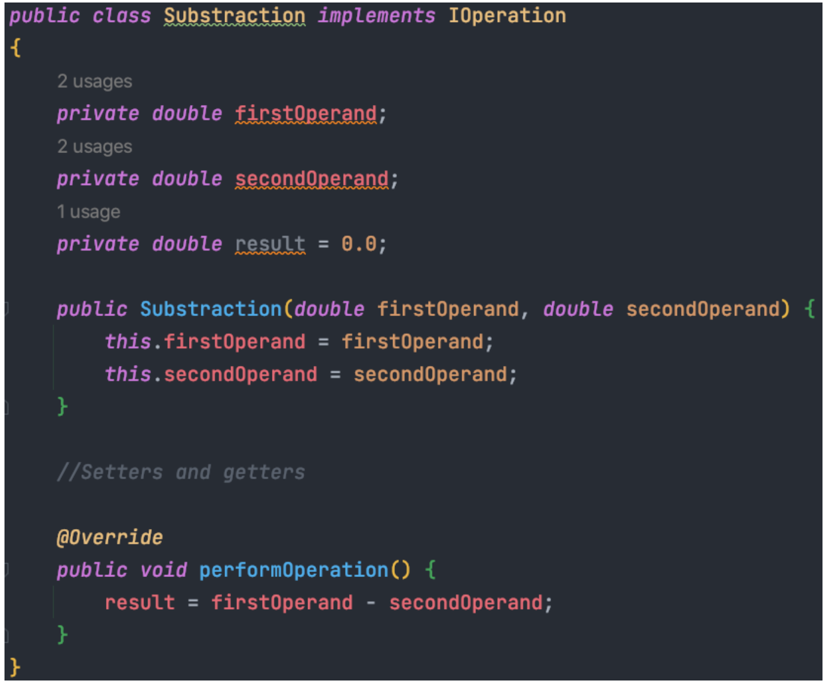
```
public class Addition implements IOperation
{
    2 usages
    private double firstOperand;
    2 usages
    private double secondOperand;
    1 usage
    private double result = 0.0;

    public Addition(double firstOperand, double secondOperand) {
        this.firstOperand = firstOperand;
        this.secondOperand = secondOperand;
    }

    //Setters and getters

    @Override
    public void performOperation() {
        result = firstOperand + secondOperand;
    }
}
```



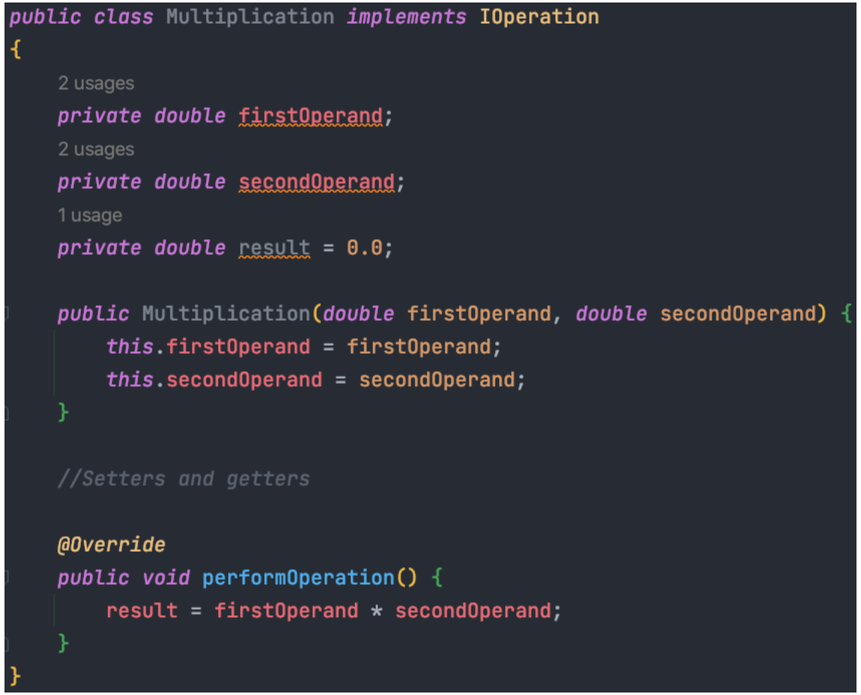
```
public interface IOperation {
    void performOperation();
}
```

PITECH PLUS

Open closed principle



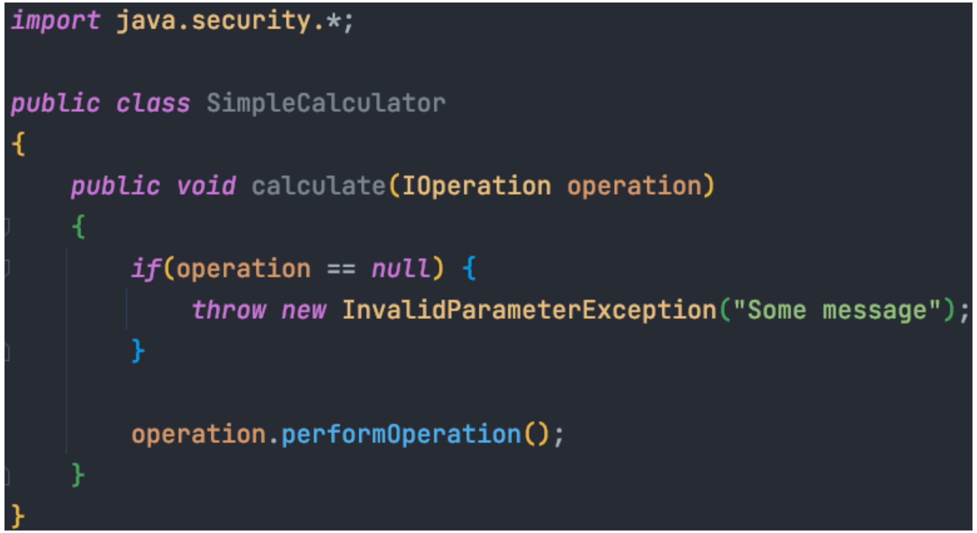
```
public class Substraction implements IOperation
{
    2 usages
    private double firstOperand;
    2 usages
    private double secondOperand;
    1 usage
    private double result = 0.0;

    public Substraction(double firstOperand, double secondOperand) {
        this.firstOperand = firstOperand;
        this.secondOperand = secondOperand;
    }

    //Setters and getters

    @Override
    public void performOperation() {
        result = firstOperand - secondOperand;
    }
}
```



```
public class Multiplication implements IOperation
{
    2 usages
    private double firstOperand;
    2 usages
    private double secondOperand;
    1 usage
    private double result = 0.0;

    public Multiplication(double firstOperand, double secondOperand) {
        this.firstOperand = firstOperand;
        this.secondOperand = secondOperand;
    }

    //Setters and getters

    @Override
    public void performOperation() {
        result = firstOperand * secondOperand;
    }
}
```

PITECH+PLUS

Open closed principle



```java
import java.security.*;

public class SimpleCalculator
{
    public void calculate(IOperation operation)
    {
        if(operation == null) {
            throw new InvalidParameterException("Some message");
        }

        operation.performOperation();
    }
}
```

PITECH+PLUS

Code smells

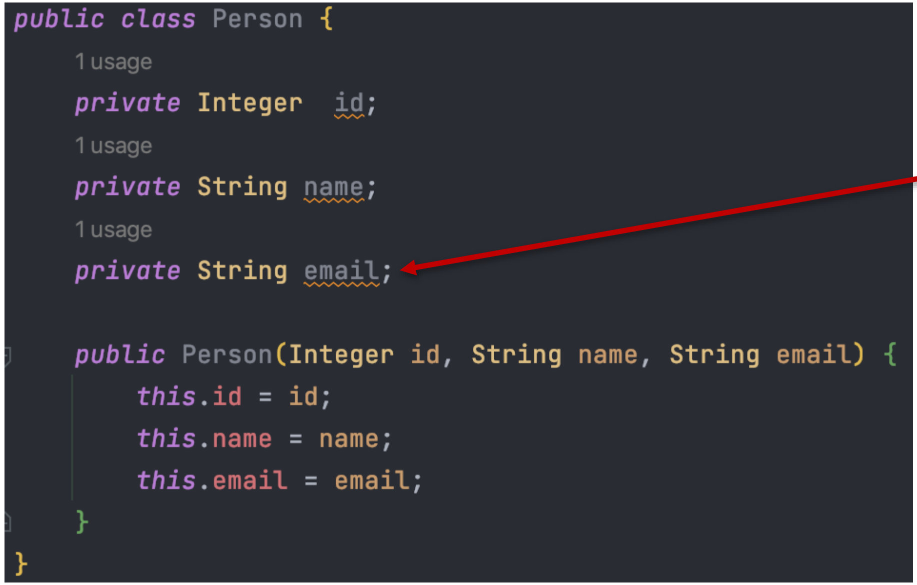
PITECH+PLUS

Code smells

A code smell is:
- easy to spot
- they are an indicator of the problem, not the problem itself

PITECH+PLUS

Code smells – **Primitive obsession**

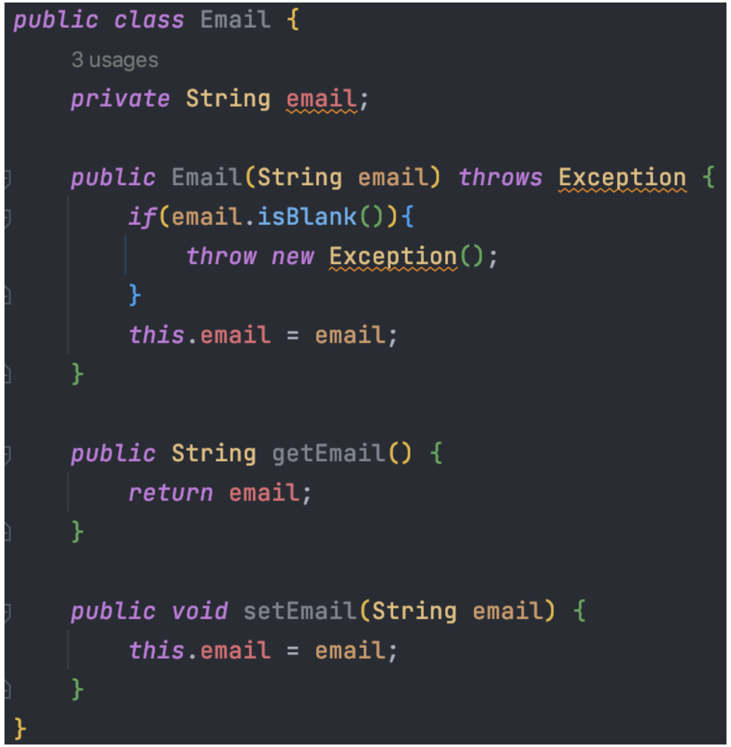


```
public class Person {
    1 usage
    private Integer  id;
    1 usage
    private String name;
    1 usage
    private String email;

    public Person(Integer id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }
}
```

Is the email just a string?

PITECH PLUS

Code smells – **Primitive obsession**



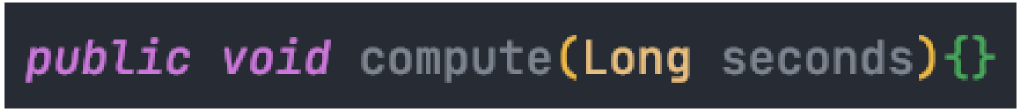
```
public class Email {
    3 usages
    private String email;

    public Email(String email) throws Exception {
        if(email.isBlank()){
            throw new Exception();
        }
        this.email = email;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

PITECH PLUS

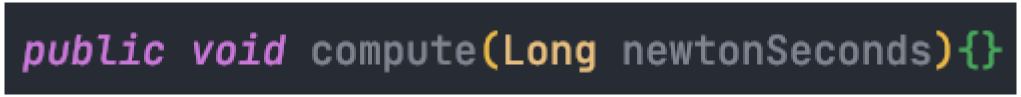
Code smells – Primitive obsession

How a type error cost NASA $ 327 million

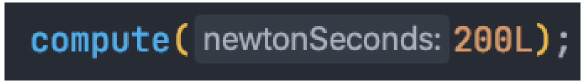
Code smells – Primitive obsession



```java
public void compute(Long seconds){}
```



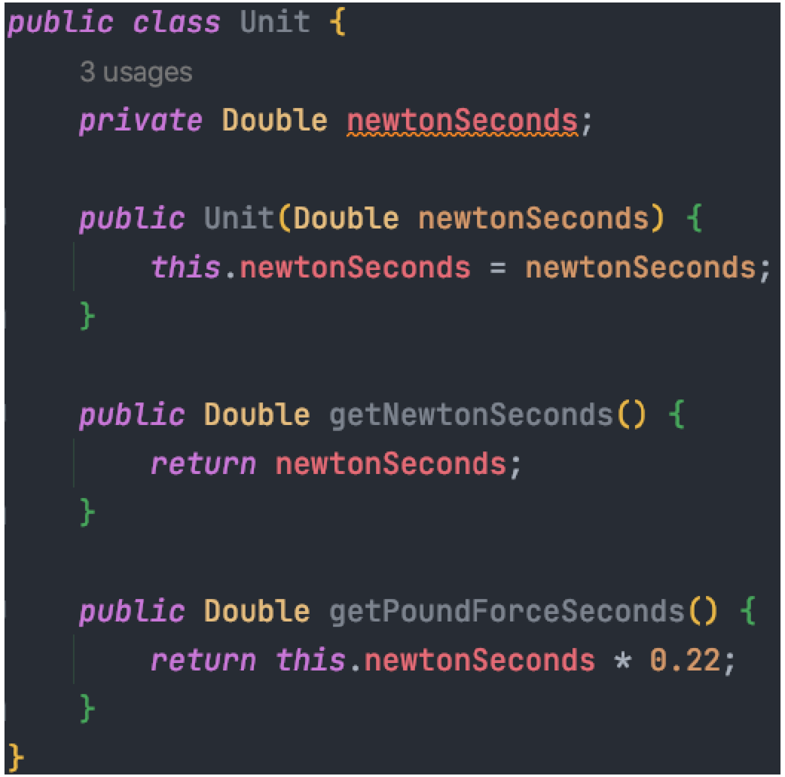
```java
public void compute(Long newtonSeconds){}
```



```java
compute( newtonSeconds: 200L);
```

PITECH+PLUS

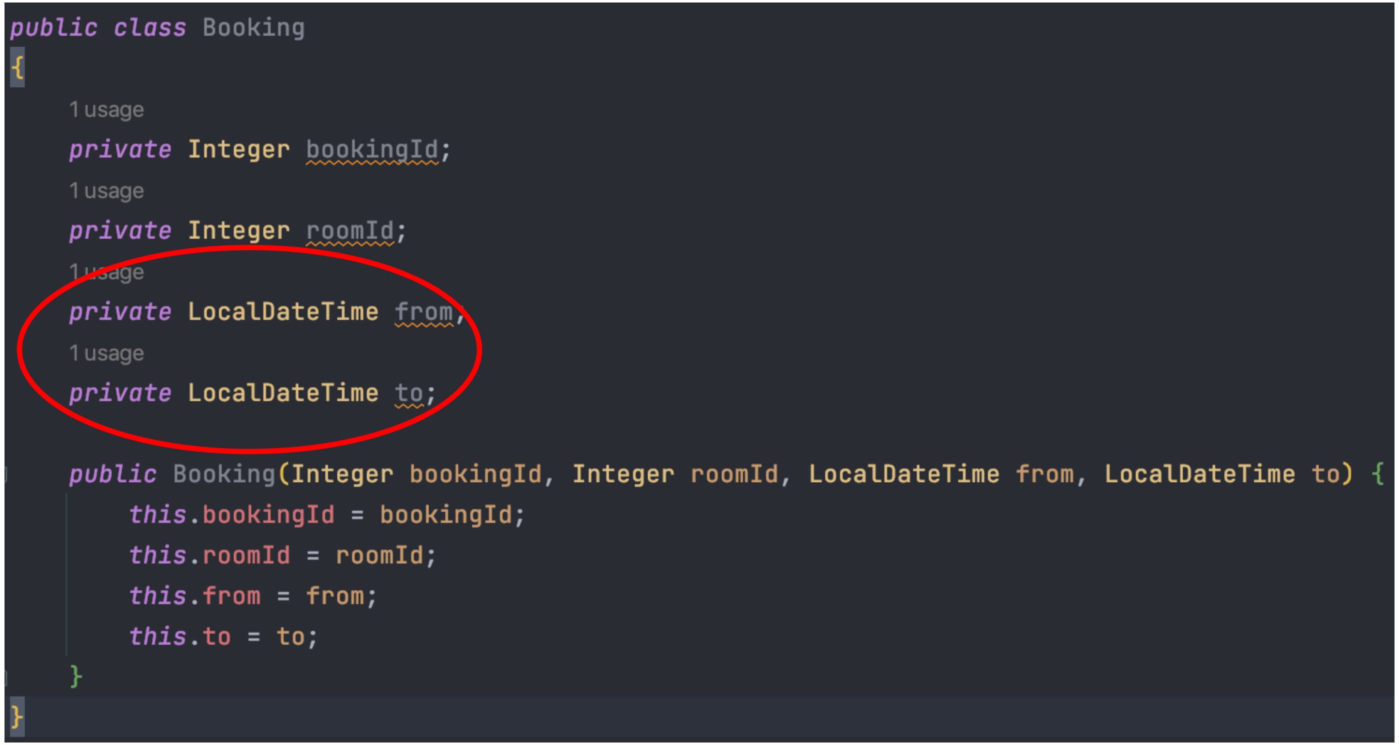
Code smells – Primitive obsession



```
public class Unit {
    3 usages
    private Double newtonSeconds;

    public Unit(Double newtonSeconds) {
        this.newtonSeconds = newtonSeconds;
    }

    public Double getNewtonSeconds() {
        return newtonSeconds;
    }

    public Double getPoundForceSeconds() {
        return this.newtonSeconds * 0.22;
    }
}
```

PITECH+PLUS

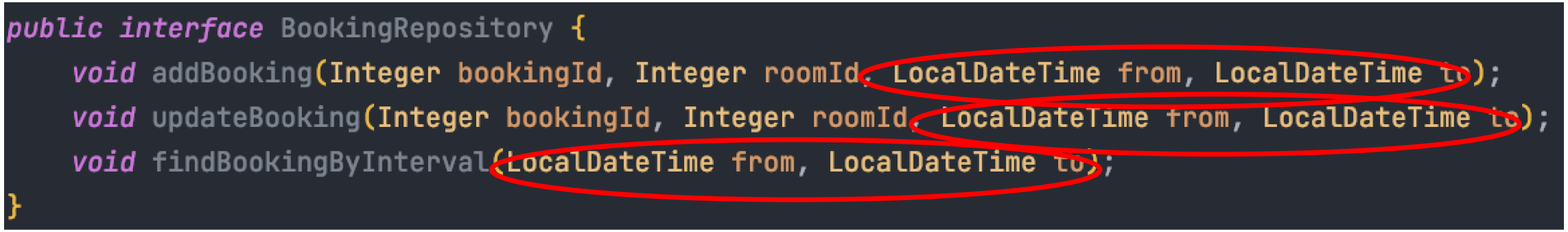
Code smells – **Data Clumps**



```
public class Booking
{
    1 usage
    private Integer bookingId;
    1 usage
    private Integer roomId;
    1 usage
    private LocalDateTime from;
    1 usage
    private LocalDateTime to;

    public Booking(Integer bookingId, Integer roomId, LocalDateTime from, LocalDateTime to) {
        this.bookingId = bookingId;
        this.roomId = roomId;
        this.from = from;
        this.to = to;
    }
}
```

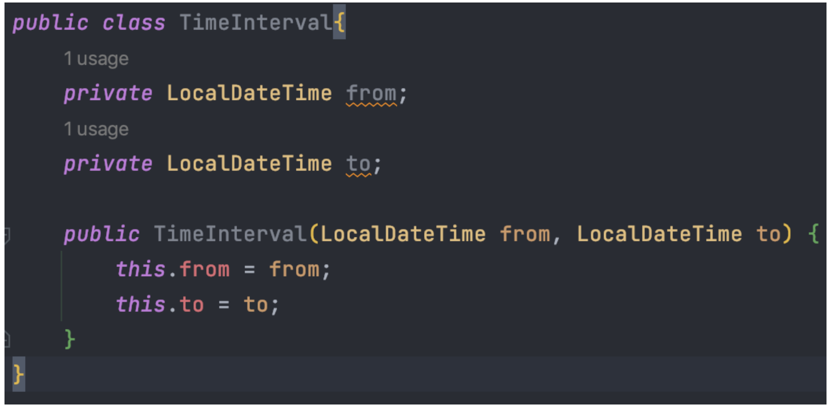
PITECH PLUS

Code smells – **Data Clumps**

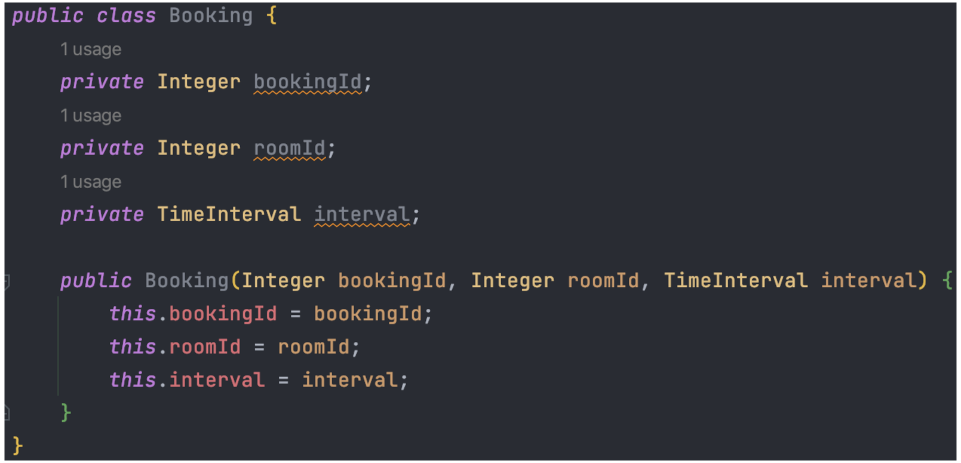


```java
public interface BookingRepository {
    void addBooking(Integer bookingId, Integer roomId, LocalDateTime from, LocalDateTime to);
    void updateBooking(Integer bookingId, Integer roomId, LocalDateTime from, LocalDateTime to);
    void findBookingByInterval(LocalDateTime from, LocalDateTime to);
}
```

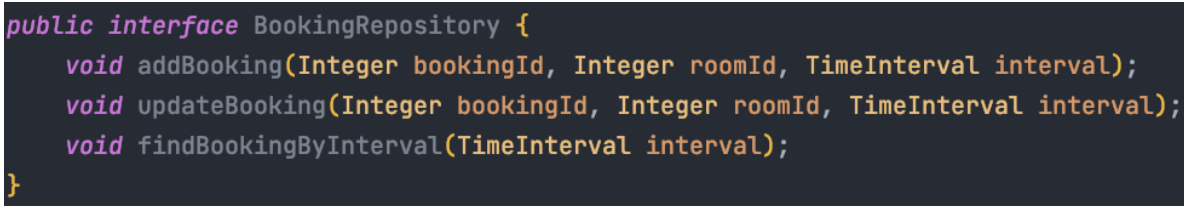
Code smells – **Data Clumps**



```java
public class TimeInterval{
    1 usage
    private LocalDateTime from;
    1 usage
    private LocalDateTime to;

    public TimeInterval(LocalDateTime from, LocalDateTime to) {
        this.from = from;
        this.to = to;
    }
}
```



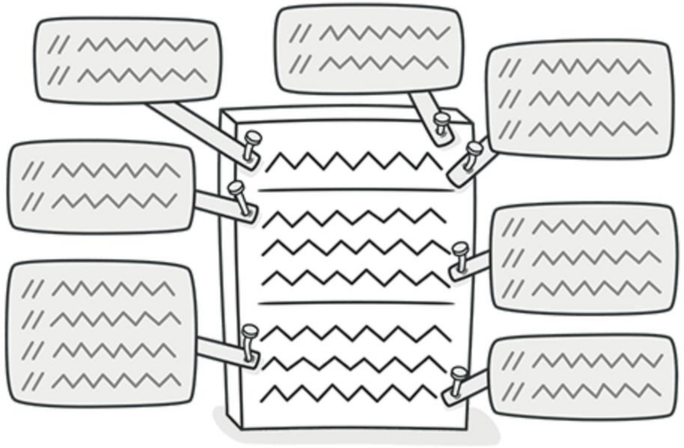
```java
public class Booking {
    1 usage
    private Integer bookingId;
    1 usage
    private Integer roomId;
    1 usage
    private TimeInterval interval;

    public Booking(Integer bookingId, Integer roomId, TimeInterval interval) {
        this.bookingId = bookingId;
        this.roomId = roomId;
        this.interval = interval;
    }
}
```



```java
public interface BookingRepository {
    void addBooking(Integer bookingId, Integer roomId, TimeInterval interval);
    void updateBooking(Integer bookingId, Integer roomId, TimeInterval interval);
    void findBookingByInterval(TimeInterval interval);
}
```

PITECH+PLUS

Code smells – **Comments**



PITECH+PLUS

Code smells – **Comments**

```
//
// Dear maintainer:
//
// Once you are done trying to 'optimize' this routine,
// and have realized what a terrible mistake that was,
// please increment the following counter as a warning
// to the next guy:
//
// total_hours_wasted_here = 42
//
```

```
/**
* For the brave souls who get this far: You are the chosen ones,
* the valiant knights of programming who toil away, without rest,
* fixing our most awful code. To you, true saviors, kings of men,
* I say this: never gonna give you up, never gonna let you down,
* never gonna run around and desert you. Never gonna make you cry,
* never gonna say goodbye. Never gonna tell a lie and hurt you.
*/
```

```
//When I wrote this, only God and I understood what I was doing
//Now, God only knows
```

```
/* This is O(scary), but seems quick enough in practice. */
```

PITECH+PLUS

Useful links and books

1. https://refactoring.guru/

2. ***Clean Code: A Handbook of Agile Software Craftsmanship*** by Robert C. Martin

3. ***Clean Architecture: A Craftsman's Guide to Software Structure and Design*** by Robert C. Martin

4. ***Head First Design Patterns: A Brain-Friendly Guide*** by Eric Freeman, Bert Bates, Kathy Sierra, Elisabeth Robson

5. ***99 Bottles of OOP*** by Sandy Metz

PITECH+PLUS

Innovation. Partnership. Evolution.

Let's build something great together!

Coratim Business Center
Campul Painii nr. 3-5,
Cluj Napoca, Romania

T. +4(0) 372 909 166
www.pitechplus.com

Facebook I Linkedin:
@PitechPlus