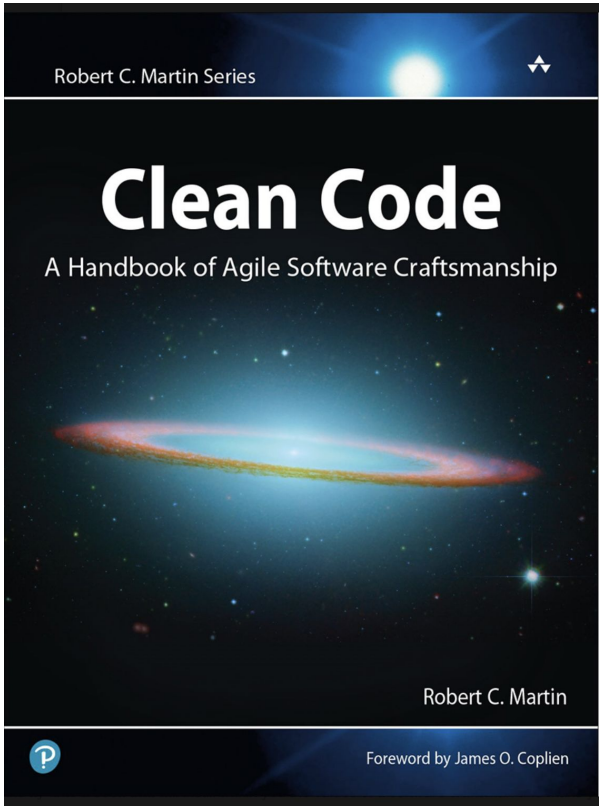
Clean Code

Jesús Salas
Software Developer in Adevinta Spain

jesus.salas.459@gmail.com

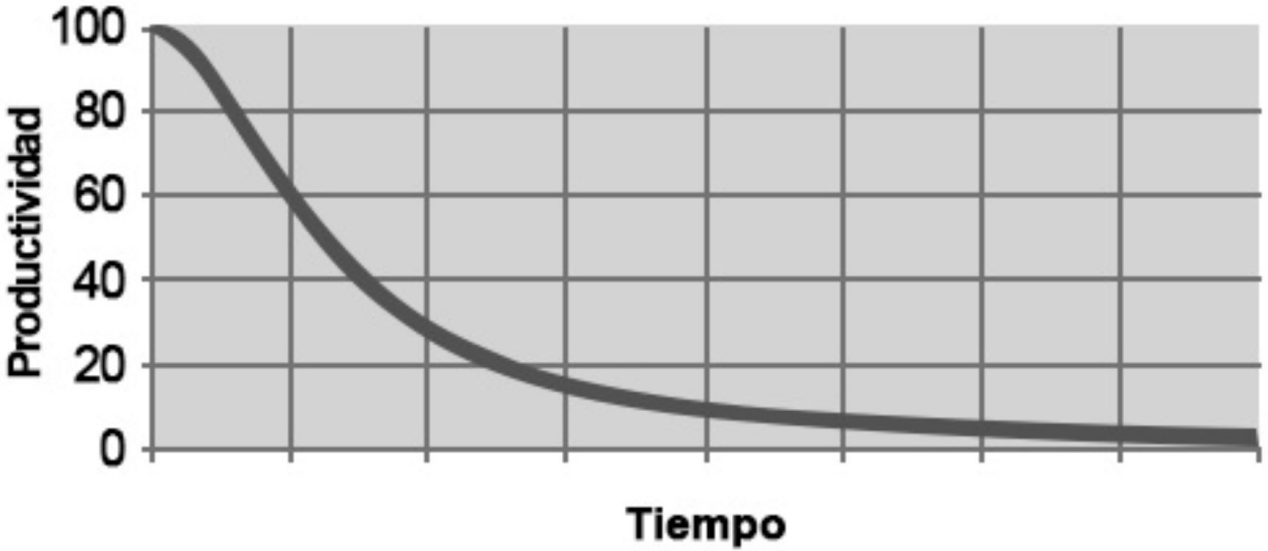
Agenda

1. Clean Code
2. Refactoring
3. Ejercicios prácticos



Robert C. Martin (Uncle Bob)
Software Engineer and Author

What is Clean Code?


100
80
60
40
20
0
Productividad
Tiempo

Best practices to achieve a clean code

OOP Recap: Classes and Objects

1. Naming
2. Methods
3. Comments
4. Format
5. Classes and objects

OOP Recap: Classes and Objects

```
public class Car {
   private int speed;
   private int fuel;

   public Car() {
       speed = 0;
       fuel = 80;
   }

   public void accelerate() {
       int speedToIncrease = 10;
       speed = speed + speedToIncrease;
   }

   public void fillTankWith(int liters) {
       fuel = fuel + liters;
   }

   public int currentSpeed() {
       return speed;
   }
}
```

```java
public class Main {
   public static void main(String[] args) {
        Car car = new Car();

        car.accelerate();
        car.accelerate();

        int fuelLiters = 30;
        car.fillTankWith(fuelLiters);

        int currentSpeed = car.currentSpeed();
        System.out.println(currentSpeed);  // 20
   }
}
```

1. Naming

Descriptive names

```java
int a = 24;
int b = 7;
int c = a * b;
System.out.println("Hours of a week: " + c);
```

```java
int hoursOfDay = 24;
int daysOfWeek = 7;
int hoursOfWeek = hoursOfDay * daysOfWeek;
System.out.println("Hours of a week: " + hoursOfWeek);
```

Avoid magic numbers

```
public void showHoursFor(int weeks) {
   int hours = weeks * 7 * 24;
   System.out.println(weeks + " weeks have " + hours + " hours");
}
```

```
final int DAYS_OF_WEEK = 7;
final int HOURS_OF_DAY = 24;

public void showHoursFor(int weeks) {
   int hours = weeks * DAYS_OF_WEEK * HOURS_OF_DAY;
   System.out.println(weeks + " weeks have " + hours + " hours");
}
```

Do not abbreviate

```java
int wh = 8;
int wd = 3;
int twh = wh * wd;
System.out.println("Worked hours: " + twh);
```

```java
int workedHoursPerDay = 8;
int workedDays = 3;
int totalWorkedHours = workedHoursPerDay * workedDays;
System.out.println("Worked hours: " + totalWorkedHours);
```

Shorten names

```java
int hoursThatIHaveWorkedInASingleDay = 8;
int totalDaysThatIHaveWorked = 3;
int totalHoursThatIHaveWorkedInAllDays =
     hoursThatIHaveWorkedInASingleDay * totalDaysThatIHaveWorked;
System.out.println("Worked hours: " + totalHoursThatIHaveWorkedInAllDays);
```

```java
int workedHoursPerDay = 8;
int workedDays = 3;
int totalWorkedHours = workedHoursPerDay * workedDays;
System.out.println("Worked hours: " + totalWorkedHours);
```

Code with business language


```
int apartments;
int homes;
```

```
int flats;
int houses;
```

2. Methods

Descriptive names

```
public int calculate(int a, int b) {
    int c = a + b;
    return c;
}
```

```
public int sum(int operand1, int operand2) {
    int total = operand1 + operand2;
    return total;
}
```

Short methods

```
public void performOperationsAndPrintResults(int operand1, int operand2) {
    int sum = operand1 + operand2;
    int substraction = operand1 - operand2;
    int multiplication = operand1 * operand2;

    System.out.println("Sum result is: " + sum);
    System.out.println("Substraction result is: " + substraction);
    System.out.println("Multiplication result is: " + multiplication);
}
```

```java
public int sum(int operand1, int operand2) {
    return operand1 + operand2;
}

public int substract(int operand1, int operand2) {
    return operand1 - operand2;
}

public int multiply(int operand1, int operand2) {
    return operand1 * operand2;
}

public void printResults(int sum, int substraction, int multiplication) {
    System.out.println("Sum result is: " + sum);
    System.out.println("Substraction result is: " + substraction);
    System.out.println("Multiplication result is: " + multiplication);
}
```

Avoid redundancy

```
public String getPhoneNumberOfCustomerId(String customerId) {
    . . .
}
```

```
public String phoneNumberOf(String customerId) {
    . . .
}
```

Express intention

```
public List<String> get(String customerId) {
    . . .
}
```

```
public List<String> mailsOf(String customerId) {
    . . .
}
```

Express intention (2)

```
public void process(String customerId, Email email) {
    . . .
}
```

```
public void sendEmailTo(String customerId, Email email) {
    . . .
}
```

Express intention (3)

```
public List<String> save(Customer customer) {
    . . .
}
```

```
public void save(Customer customer) {
    . . .
}
```

Avoid boolean arguments

```
public void sendEmailTo(String customerId, Email email,
     boolean addAttachment) {
   . . .
}
```

```
public void sendEmailTo(String customerId, Email email) {
   . . .
}

public void sendEmailWithAttachmentTo(String customerId, Email email) {
   . . .
}
```

Fewer arguments

```
public void save(String customerId, String email, String phone,
      String address) {
    . . .
}
```

```
public void save(Customer customer) {
    . . .
}
```

Avoid side effects

```
public String addressOf(String customerId) {
    . . .
    database.update(customerId)
    . . .
}
```

Avoid side effects (2)

```
public Car cheaperOf(Car car1, Car car2) {
    Car cheaperCar;
    int discount = 1000;

    car1.setPrice(car1.price() - discount);

    if (car1.price() < car2.price()) {
        cheaperCar = car1;
    } else {
        cheaperCar = car2;
    }

    return cheaperCar;
}
```

Distribute responsability

```
public static void main(String[] args) {

    . . .

    System.out.println("Enter book id: ");
    String bookId = scanner.nextLine();

    System.out.println("Enter discount to apply: ");
    int discount = parseInt(scanner.nextLine());

    applyDiscountToBookAndSaveChanges(bookId, discount);


    . . .
}
```

```
private void applyDiscountToBookAndSaveChanges(String bookId, int discount) {

    . . .

    Book book = database.findBy(bookdId);
    int newPrice = book.price() - book.price() / 100 * discount;
    book.setPrice(newPrice);
    database.save(book);

    . . .
}
```

```
public static void main(String[] args) {

    . . .

    System.out.println("Enter book id: ");
    String bookId = scanner.nextLine();

    System.out.println("Enter discount to apply: ");
    int discount = parseInt(scanner.nextLine());

    Book book = findBy(bookId);
    book.apply(discount);
    save(book);

    . . .
}
```

```
private Book findBy(String bookdId) {
   return database.findBy(bookdId);
}

private void save(Book book) {
   return database.save(book);
}

class Book {
   . . .
   private int price;
   public void apply(int discount) {
      this.price = price - price / 100 * discount;
   }
   . . .
}
```

3. Comments

Avoid comments

```
// Given two cars returns the one with the lower price
public Car calculate(Car car1, Car car2) {
    Car result;

    // Compare prices
    if (car1.pr() < car2.pr()) {
        result = car1;
    } else {
        result = car2;
    }

    return result;
}
```

```
public Car cheaperOf(Car car1, Car car2) {
   Car cheaperCar;

   if (car1.price() < car2.price()) {
       cheaperCar = car1;
   } else {
       cheaperCar = car2;
   }

   return cheaperCar;
}
```

Problem: Redundancy

```
public class Car {
    // Current price
    private int price;
    // Car license plate
    private String plate;
    // Car color
    private String color;

    // Constructor
    public Car(int price, String plate, String color) {
        this.price = price;
        this.plate = plate;
        this.color = color;
    }
    ...
}
```

```java
public class Car {
    private int price;
    private String plate;
    private String color;

    public Car(int price, String plate, String color) {
        this.price = price;
        this.plate = plate;
        this.color = color;
    }
    ...
}
```

Problem: Lack of maintenance

```
// Given two cars returns the one with the lower price
public Car mostExpensiveOf(Car car1, Car car2) {
   Car mostExpensiveCar;

   if (car1.price() > car2.price()) {
       mostExpensiveCar = car1;
   } else {
       mostExpensiveCar = car2;
   }

   return mostExpensiveCar;
}
```

Avoid change logs

```
// Change log:
//    20/03/2017: Added color feature
//    12/05/2018: Added gps feature
public class Car {
   private int price;
   private String plate;
   private String color;
   private boolean gps;

   public Car(int price, String plate, String color, boolean gps) {
       this.price = price;
       this.plate = plate;
       this.color = color;
       this.gps = gps;
   }
}
```

Avoid commented code

```
public class Car {
    private int price;
    private String plate;
    private String color;
    //private boolean gps;

    //public Car(int price, String plate, String color, boolean gps) {
    public Car(int price, String plate, String color) {
        this.price = price;
        this.plate = plate;
        this.color = color;
        //this.gps = gps;
    }
}
```

Remove dead code

```
public class Car {
    private int speed = 0;

    public void accelerate() {
        speed = speed + 10;
    }

    public void stop() {
        speed = 0;
    }
}
```

```
public class Car {
    private int speed = 0;

    public void accelerate() {
        speed = speed + 10;
    }
}
```

4. Formatting

Identation

```
public int divide(int dividend, int divisor) {

if (divisor == 0) {

throw new InvalidOperationException("Cannot divide by zero");

}

return dividend / divisor;

}
```

```
public int divide(int dividend, int divisor) {
    if (divisor == 0) {
        throw new InvalidOperationException("Cannot divide by zero");
    }
    return dividend / divisor;
}
```

Identation (2)

```
public int maxOf(int number1, int number2, int number3) {
     if (number1 >= number2) {
   if (number1 >= number3) {
        return number1;
       } else {
           return number3;
     }
         } else {
    if (number2 >= number3) {
           return number2;
 } else {
                return number3;
       }
}
}
```

```java
public int maxOf(int number1, int number2, int number3) {
    if (number1 >= number2) {
        if (number1 >= number3) {
            return number1;
        } else {
            return number3;
        }
    } else {
        if (number2 >= number3) {
            return number2;
        } else {
            return number3;
        }
    }
}
```

Spacing (horizontal)

```
public void apply(int discount) {
   this.price=price-price/100*discount;
}
```

```
public void apply (int discount) {
   this.price = price - price / 100 * discount;
}
```

Spacing (vertical)

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int number1, number2, number3, average;
    System.out.println("Enter first number: ");
    number1 = scanner.nextInt();
    System.out.println("Enter second number: ");
    number2 = scanner.nextInt();
    System.out.println("Enter third number: ");
    number3 = scanner.nextInt();
    average = (number1 + number2 + number3) / 3;
    System.out.println("Average is: " + average);
}
```

Spacing (vertical)

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int number1, number2, number3, average;

    System.out.println("Enter first number: ");
    number1 = scanner.nextInt();
    System.out.println("Enter second number: ");
    number2 = scanner.nextInt();
    System.out.println("Enter third number: ");
    number3 = scanner.nextInt();

    average = (number1 + number2 + number3) / 3;
    System.out.println("Average is: " + average);
}
```

Good practices

- Consistency throughout the project
- Team agreements
- Code style schemes

4. Classes and Objects

Single responsability

```
public class Utils {

    public int sum(int operand1, int operand2) {

        return operand1 + operand2;

    }


    public int squareArea(int sideLength) {

        return sideLength * sideLength;

    }

}
```

```java
public class Calculator {
    public int sum(int operand1, int operand2) {
        return operand1 + operand2;
    }
}

public class Square {
    public int area(int sideLength) {
        return sideLength * sideLength;
    }
}
```

Law of Demeter

```
public class Customer {

   private String tradeName;

   private Address address;


   public Address getAddress() {

       return address;

   }

}
```

```
public class Address {

    private String streetName;

    private int streetNumber;


    public String getStreetName() {

        return streetName;

    }


    public int getStreetNumber() {

        return streetNumber;

    }

}
```

```
public static void main(String[] args) {

    Customer customer = new Customer();

    String streetName = customer.getAddress().getStreetName();

    . . .

}
```

```
public class Customer {

   private String tradeName;

   private Address address;


   public String getStreetName() {

       return address.getStreetName();

   }

}
```

```
public static void main(String[] args) {

    Customer customer = new Customer();

    String streetName = customer.getStreetName();

    . . .

}
```

What is Refactoring?

When and how to refactor

The boy scout rule

Leave the campground cleaner
than you found it.

Tips:

- DRY Principle (Don’t Repeat Yourself) But… ¡Be careful!
- Do you understand your code? And your team mates?
- Use IDE refactoring tools


THANK
YOU!


WHERE
WHAT
WHEN
WHY
HOW
WHO

https://github.com/jesus-salas-j/clean-code