Refactoring to Bind It All Together

Zoran Horvat
CEO AT CODING HELMET

@zoranh75 http://codinghelmet.com

```java
public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```

Document number
(typed in by a man)

123456-1

Control digit
(added by software)

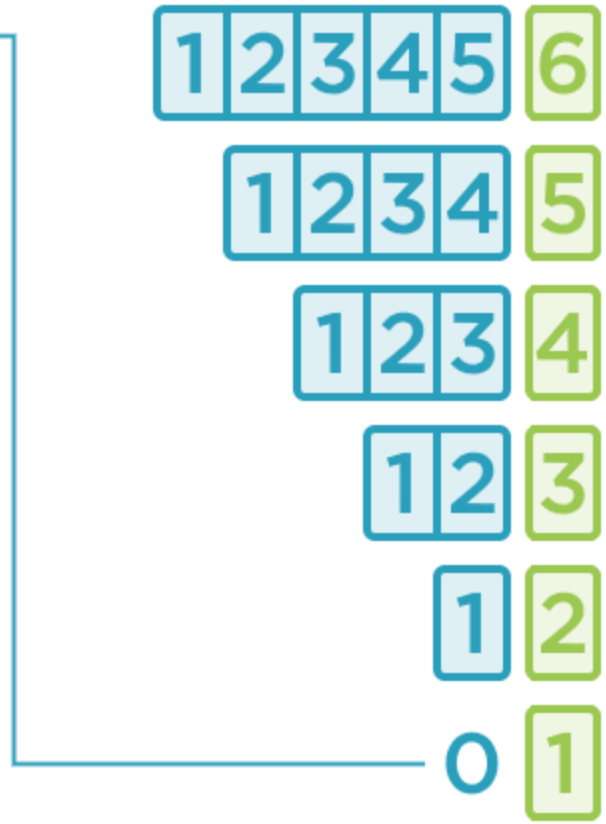
```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```

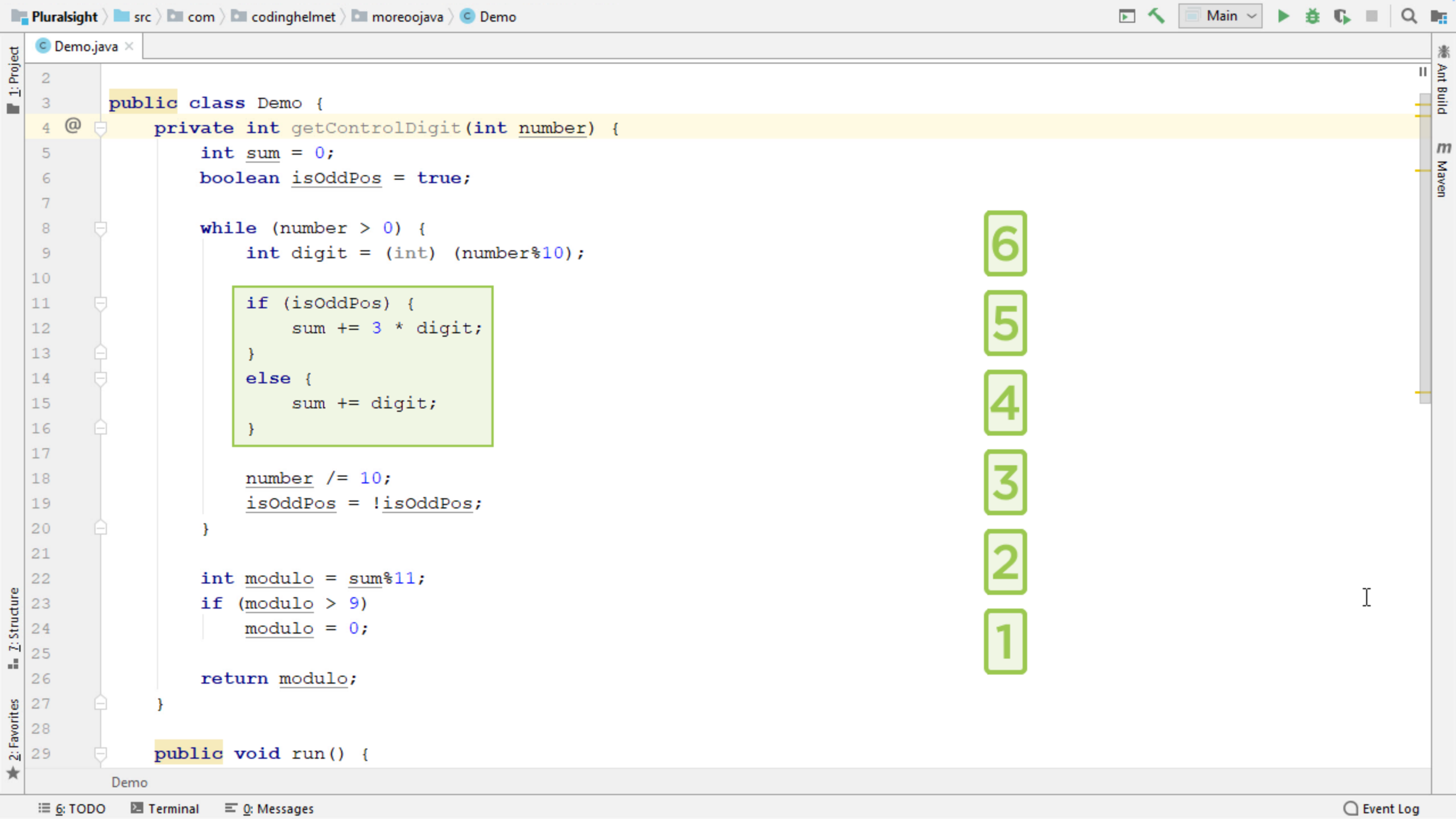
```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```

```java

public class Demo {
    @ private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```

```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```

```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```



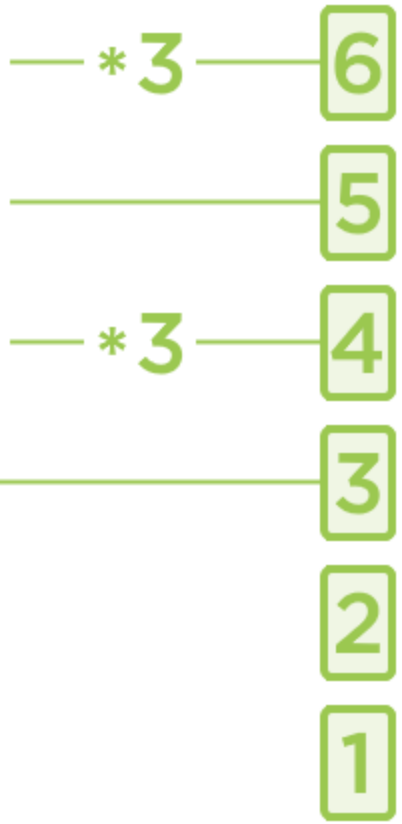

Pluralsight
src
com
codinghelmet
moreoojava
Demo
Main
Demo.java
1: Project
Ant Build
Maven
7: Structure
2: Favorites
public class Demo {
private int getControlDigit(int number) {
int sum = 0;
boolean isOddPos = true;
while (number > 0) {
int digit = (int) (number%10);
if (isOddPos) {
sum += 3 * digit;
}
else {
sum += digit;
}
number /= 10;
isOddPos = !isOddPos;
}
int modulo = sum%11;
if (modulo > 9)
modulo = 0;
return modulo;
}
public void run() {
6
5
4
3
2
1
Demo
6: TODO
Terminal
0: Messages
Event Log

```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```

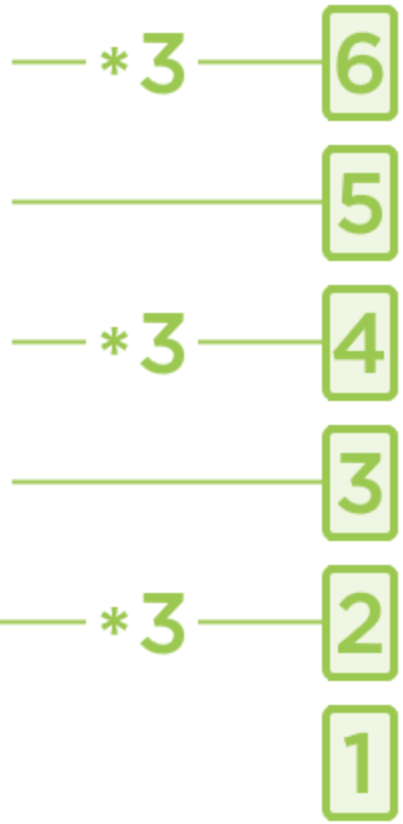
```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```

```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```

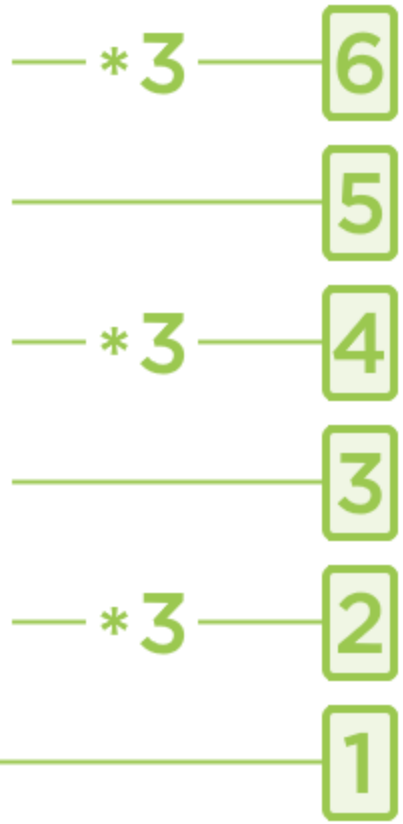
```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```



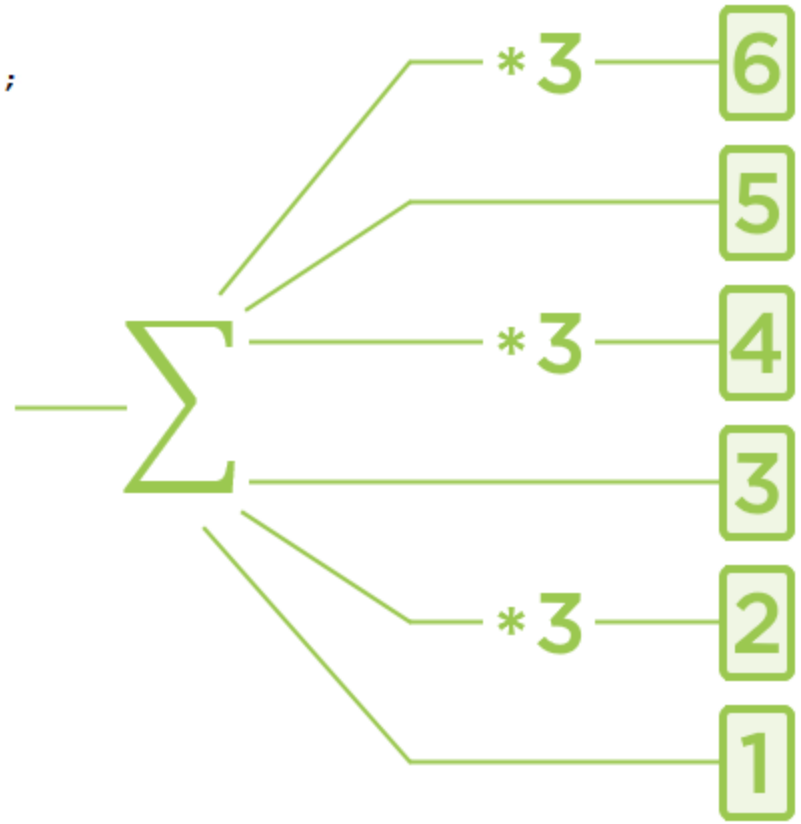
```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```



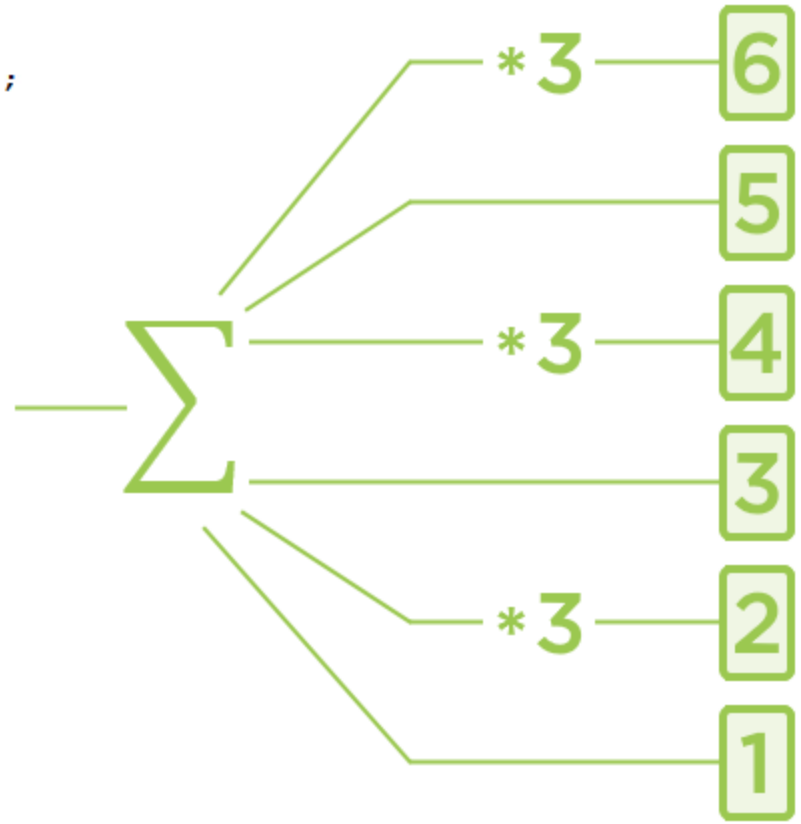
```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```



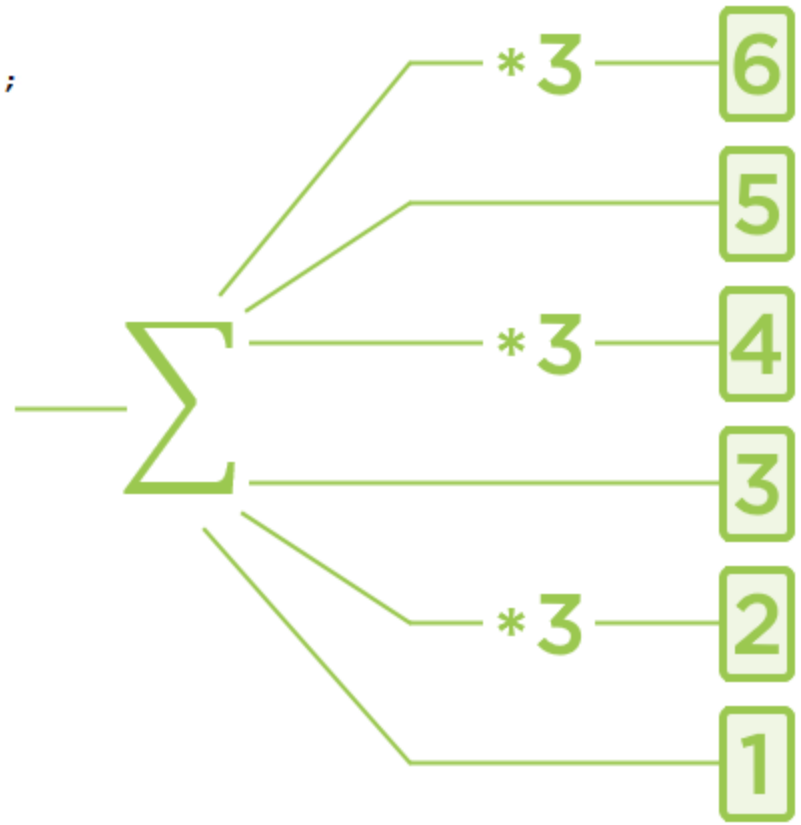
```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```



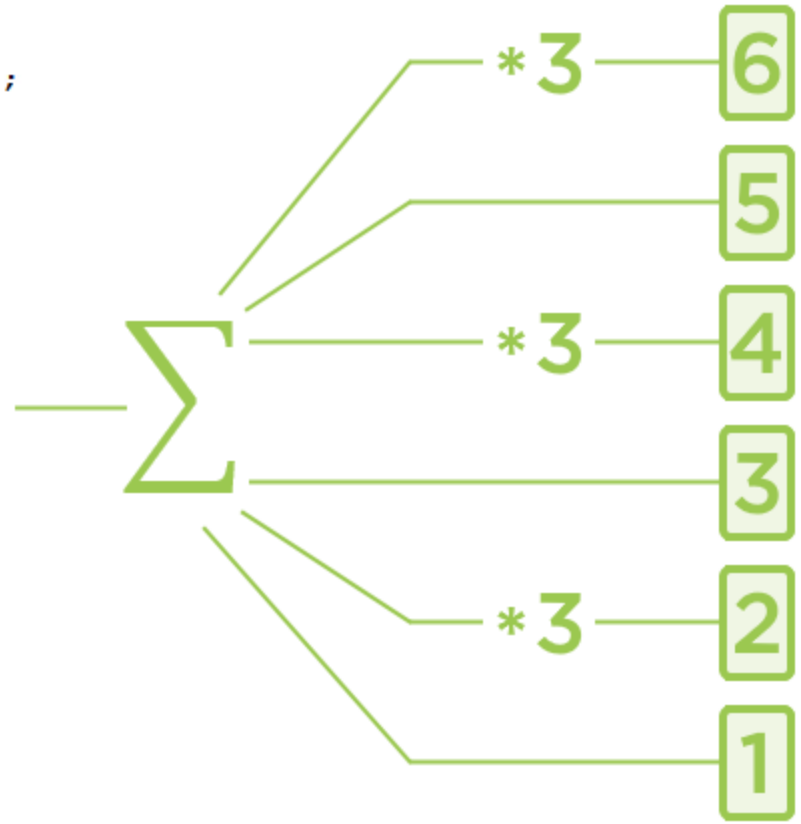
```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```



```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```



```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```

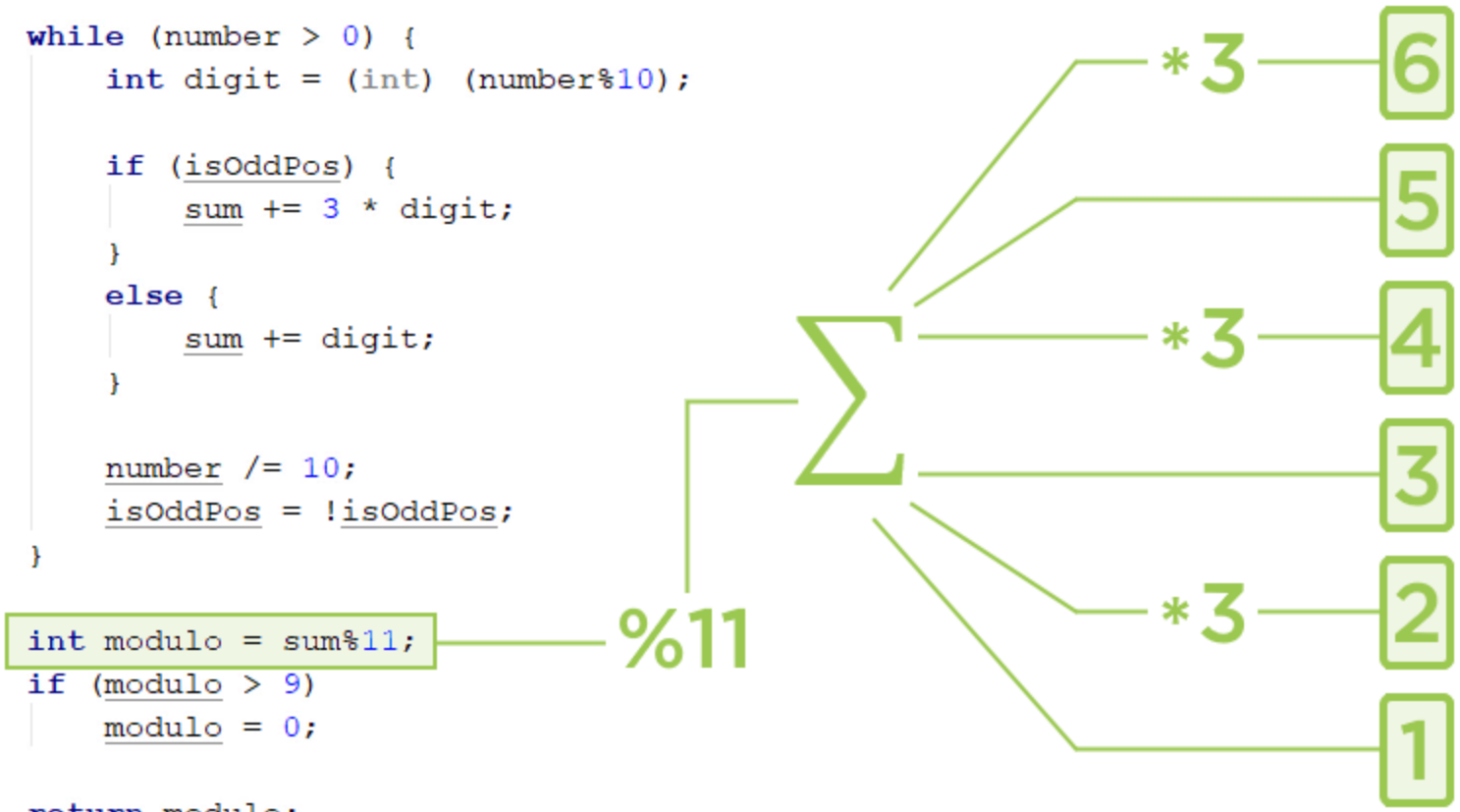


Pluralsight › src › com › codinghelmet › moreoojava › Demo

Main

Demo.java

```
2
3   public class Demo {
4       private int getControlDigit(int number) {
5           int sum = 0;
6           boolean isOddPos = true;
7
8           while (number > 0) {
9               int digit = (int) (number%10);
10
11              if (isOddPos) {
12                  sum += 3 * digit;
13              }
14              else {
15                  sum += digit;
16              }
17
18              number /= 10;
19              isOddPos = !isOddPos;
20          }
21
22          int modulo = sum%11;
23          if (modulo > 9)
24              modulo = 0;
25
26          return modulo;
27      }
28
29      public void run() {
```



Demo

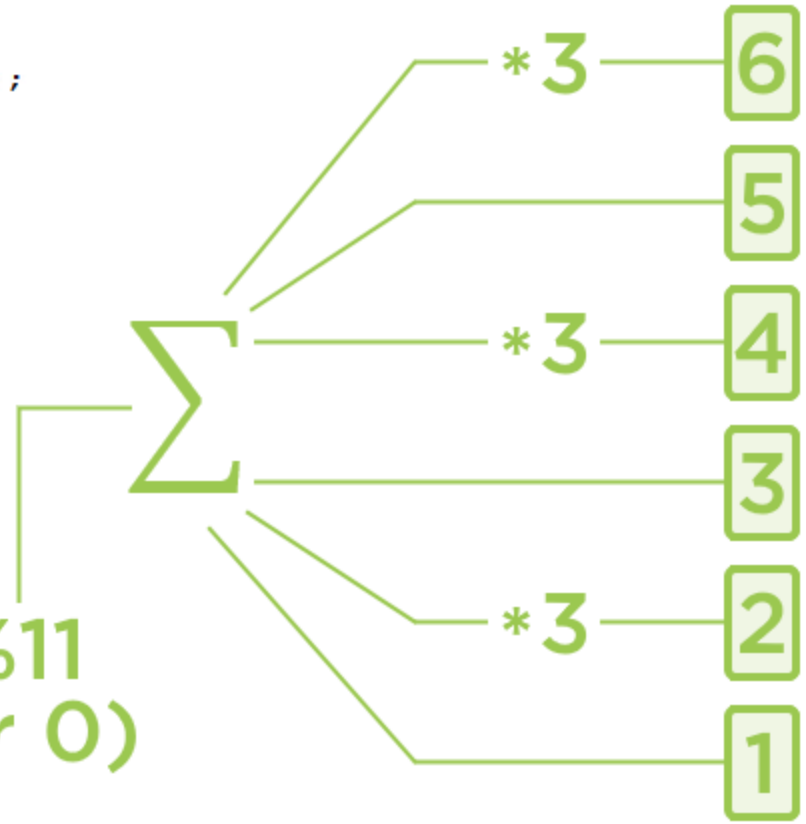
1: Project
7: Structure
2: Favorites
Ant Build
Maven
6: TODO
Terminal
0: Messages
Event Log

Pluralsight › src › com › codinghelmet › moreoojava › Demo

Main

Demo.java

```java

public class Demo {
    private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```



Demo

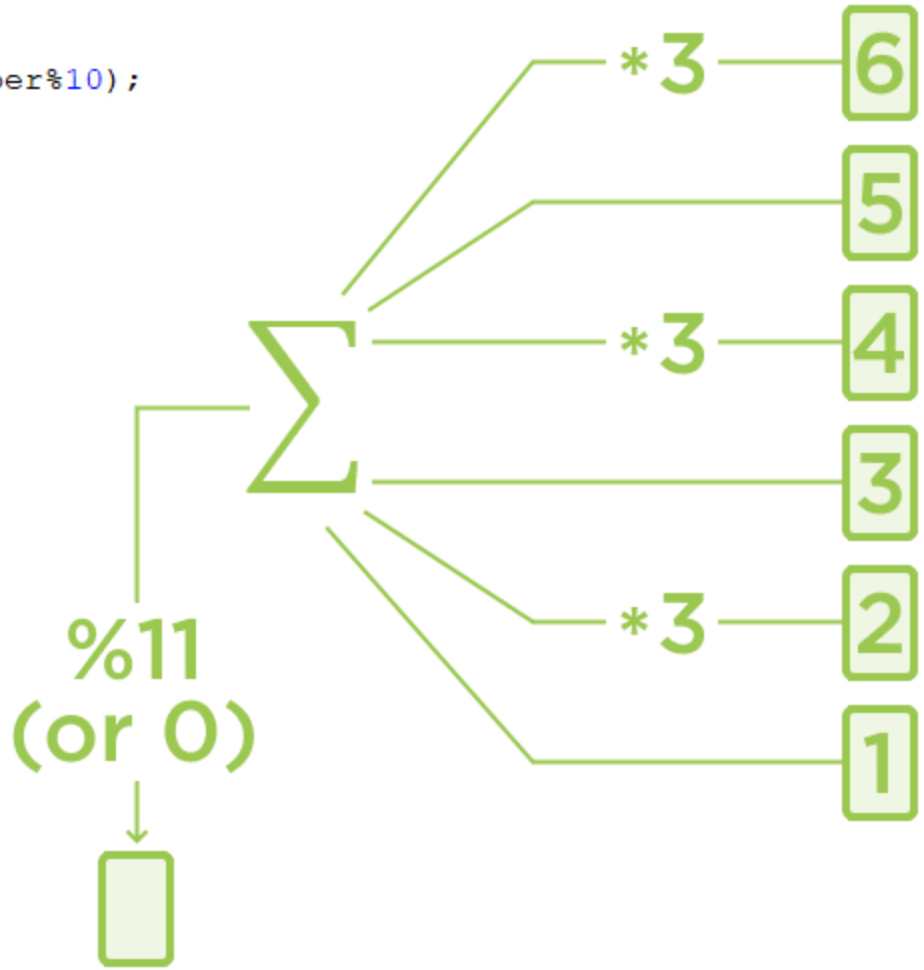
6: TODO Terminal 0: Messages Event Log

Pluralsight › src › com › codinghelmet › moreoojava › Demo

Main

Demo.java

```
public class Demo {
    @ private int getControlDigit(int number) {
        int sum = 0;
        boolean isOddPos = true;

        while (number > 0) {
            int digit = (int) (number%10);

            if (isOddPos) {
                sum += 3 * digit;
            }
            else {
                sum += digit;
            }

            number /= 10;
            isOddPos = !isOddPos;
        }

        int modulo = sum%11;
        if (modulo > 9)
            modulo = 0;

        return modulo;
    }

    public void run() {
```



Demo

6: TODO Terminal 0: Messages Event Log

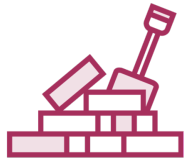
1: Project 2: Favorites 7: Structure Ant Build Maven

Assessing Code Quality

```
private int getControlDigit(int number) {
    int sum = 0;
    boolean isOddPos = true;

    while (number > 0) {
        int digit = (int) (number%10);

        if (isOddPos) { sum += 3 * digit; }
        else { sum += digit; }

        number /= 10;
        isOddPos = !isOddPos;
    }

    int modulo = sum%11;
    if (modulo > 9)
        modulo = 0;

    return modulo;
}
```

Nice?

Short?

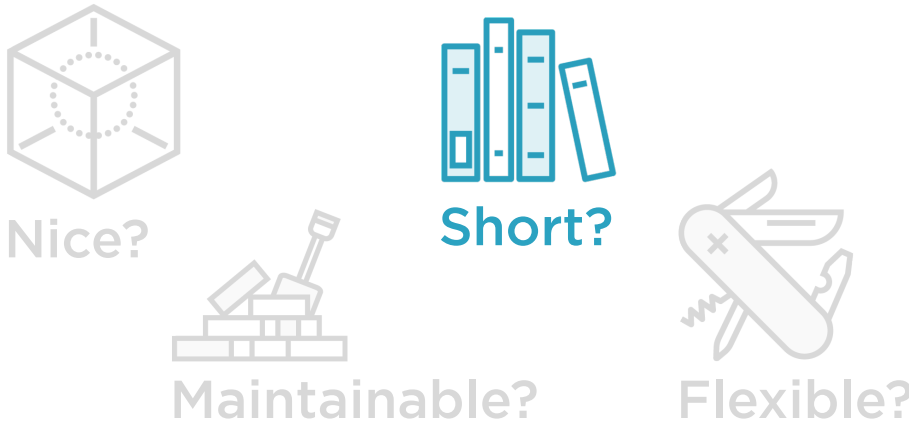


Maintainable?

Flexible?

Assessing Code Quality

```
private int getControlDigit(int number) {
    int sum = 0;
    boolean isOddPos = true;

    while (number > 0) {
        int digit = (int) (number%10);

        if (isOddPos) { sum += 3 * digit; }
        else { sum += digit; }

        number /= 10;
        isOddPos = !isOddPos;
    }

    int modulo = sum%11;
    if (modulo > 9)
        modulo = 0;

    return modulo;
}
```



Complex requirements lead to longer implementation

Assessing Code Quality

```
private int getControlDigit(int number) {
    int sum = 0;
    boolean isOddPos = true;

    while (number > 0) {
        int digit = (int) (number%10);

        if (isOddPos) { sum += 3 * digit; }
        else { sum += digit; }

        number /= 10;
        isOddPos = !isOddPos;
    }

    int modulo = sum%11;
    if (modulo > 9)
        modulo = 0;

    return modulo;
}
```


Nice?

Short?

Maintainable?

Flexible?

Requirements:

Multiply every other digit by three
starting with the first digit on the right
sum the results up
take modulo eleven of the sum
or substitute zero

Implementation:

Loop while not zero, take modulo 10, etc.

Assessing Code Quality

```
private int getControlDigit(int number) {
    int sum = 0;
    boolean isOddPos = true;

    while (number > 0) {
        int digit = (int) (number%10);

        if (isOddPos) { sum += 3 * digit; }
        else { sum += digit; }

        number /= 10;
        isOddPos = !isOddPos;
    }

    int modulo = sum%11;
    if (modulo > 9)
        modulo = 0;

    return modulo;
}
```


Nice?

Short?

Maintainable?

Flexible?

Procedural code doesn't match the requirements in a spoken language

What shall we do when requirements change?

Cognitive mapping

Assessing Code Quality

```
private int getControlDigit(int number) {
    int sum = 0;
    boolean isOddPos = true;

    while (number > 0) {
        int digit = (int) (number%10);

        if (isOddPos) { sum += 3 * digit; }
        else { sum += digit; }

        number /= 10;
        isOddPos = !isOddPos;
    }

    int modulo = sum%11;
    if (modulo > 9)
        modulo = 0;

    return modulo;
}
```


Nice?

Short?

Maintainable?

Flexible?

Requirements follow
logic of a man

Implementation follows
logic of a ***programming language***

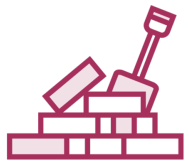
Cognitive mapping

Assessing Code Quality

```
private int getControlDigit(int number) {
    int sum = 0;
    boolean isOddPos = true;

    while (number > 0) {
        int digit = (int) (number%10);

        if (isOddPos) { sum += 3 * digit; }
        else { sum += digit; }

        number /= 10;
        isOddPos = !isOddPos;
    }

    int modulo = sum%11;
    if (modulo > 9)
        modulo = 0;

    return modulo;
}
```


Nice?

Short?



Maintainable?

Flexible?

Cognitive mapping

Added requirement:

Read numbers left-to-right

Implementation:

No substitute for the % operator!

Assessing Code Quality

```
private int getControlDigit(int number) {
    int sum = 0;
    boolean isOddPos = true;

    while (number > 0) {
        int digit = (int) (number%10);

        if (isOddPos) { sum += 3 * digit; }
        else { sum += digit; }

        number /= 10;
        isOddPos = !isOddPos;
    }

    int modulo = sum%11;
    if (modulo > 9)
        modulo = 0;

    return modulo;
}
```

Right-to-left or left-to-right

Multiplier depends on date

One modulo or the other

With or without substitute

...

...

Nice?

Short?

Maintainable?

Flexible?

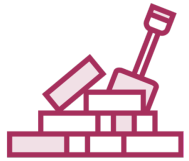
Cognitive mapping

Assessing Code Quality

```
private int getControlDigit(int number) {
    int sum = 0;
    boolean isOddPos = true;

    while (number > 0) {
        int digit = (int) (number%10);

        if (isOddPos) { sum += 3 * digit; }
        else { sum += digit; }

        number /= 10;
        isOddPos = !isOddPos;
    }

    int modulo = sum%11;
    if (modulo > 9)
        modulo = 0;

    return modulo;
}
```

Nice?

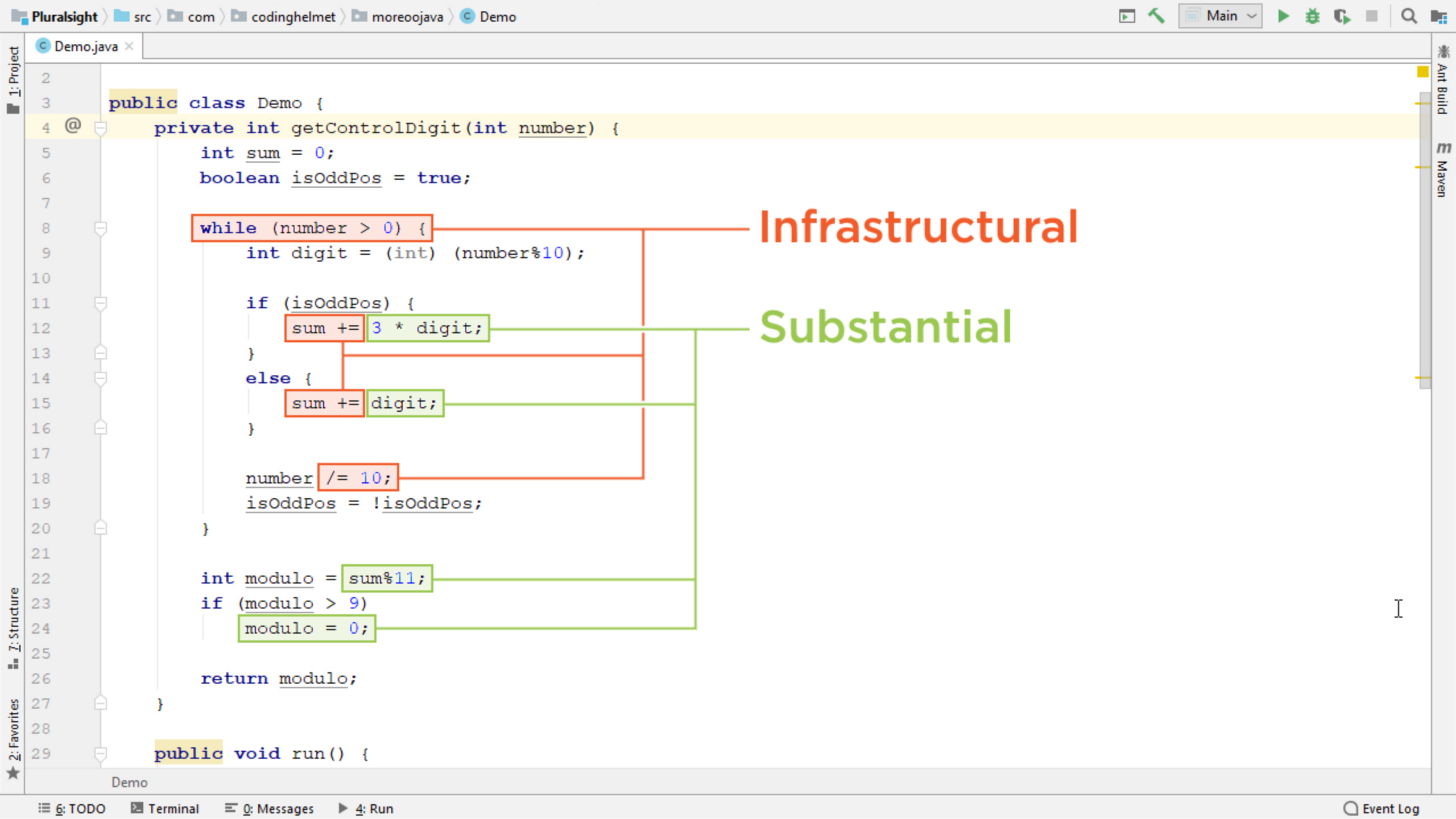
Short?



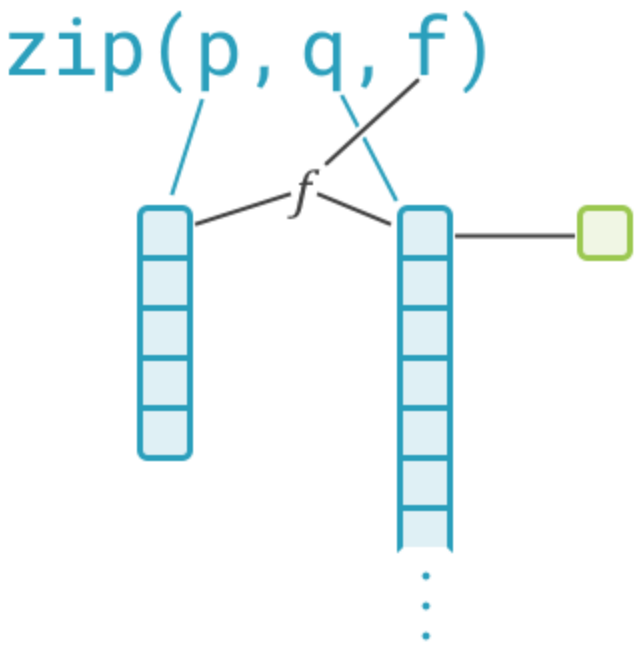
Maintainable?

Flexible?

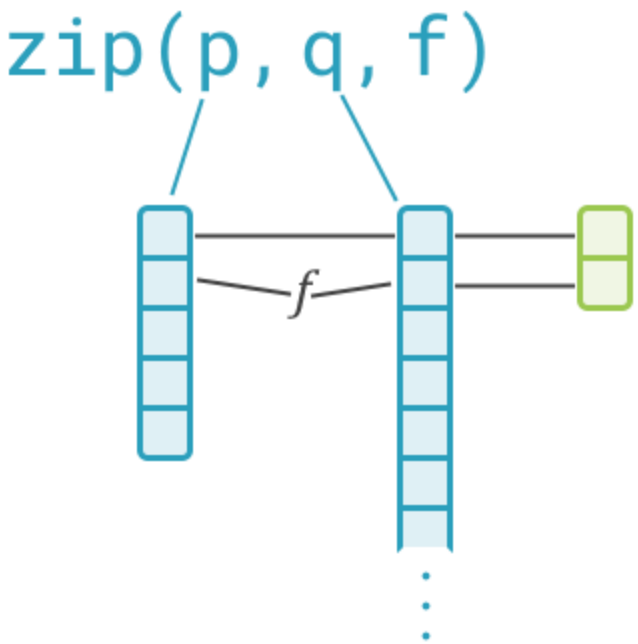
Cognitive mapping


Pluralsight
src
com
codinghelmet
moreoojava
Demo
Main
Demo.java
1: Project
Ant Build
Maven
7: Structure
2: Favorites
public class Demo {
private int getControlDigit(int number) {
int sum = 0;
boolean isOddPos = true;
while (number > 0) {
int digit = (int) (number%10);
if (isOddPos) {
sum += 3 * digit;
}
else {
sum += digit;
}
number /= 10;
isOddPos = !isOddPos;
}
int modulo = sum%11;
if (modulo > 9)
modulo = 0;
return modulo;
}
public void run() {
Infrastructural
Substantial
Demo
6: TODO
Terminal
0: Messages
4: Run
Event Log

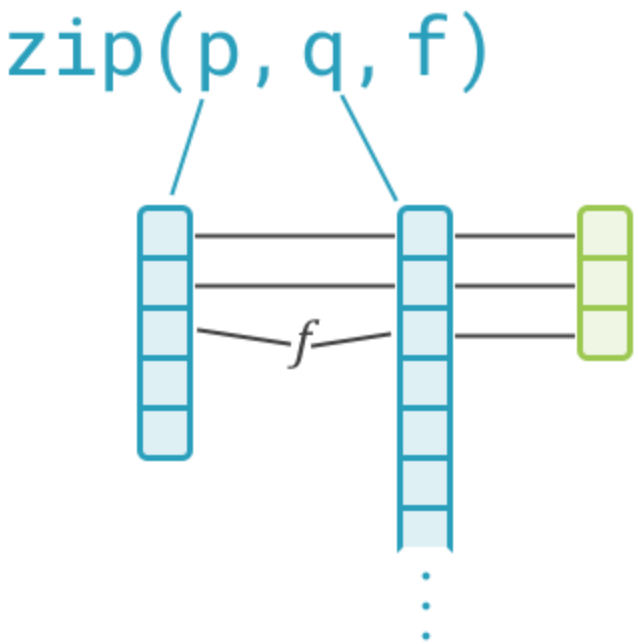
```java
public class DigitsStream implements ForwardingStream<Integer> {
    private Stream<Integer> stream;

    @Override
    public Stream<Integer> getStream() { return this.stream; }

    private DigitsStream(Stream<Integer> stream) {
        this.stream = stream;
    }

    public static DigitsStream of(Stream<Integer> stream) {
        return new DigitsStream(stream);
    }

    public int multiplyWith(int... factors) {
        return this.getStream().zip(this.repeatEndlessly(factors))
    }

    private Stream<Integer> repeatEndlessly(int... factors) {
        return Stream
            .iterate(factors, UnaryOperator.identity())
            .flatMapToInt(Arrays::stream)
            .boxed();
    }
}
```



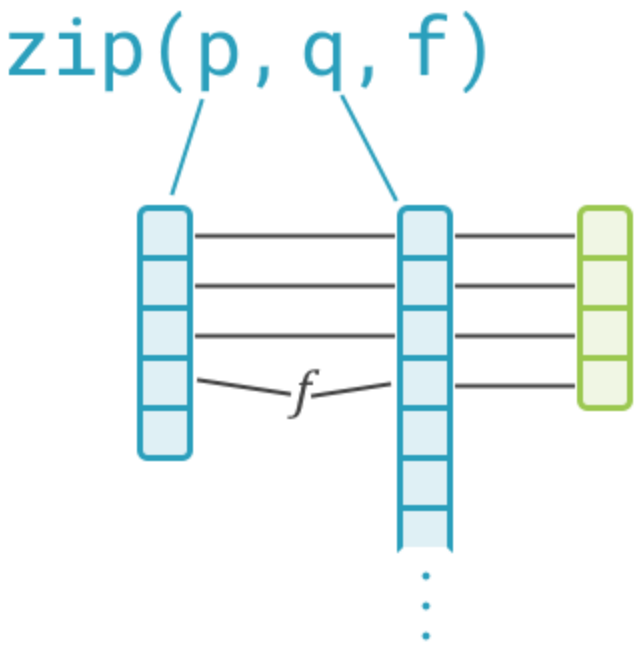
```java
public class DigitsStream implements ForwardingStream<Integer> {
    private Stream<Integer> stream;

    @Override
    public Stream<Integer> getStream() { return this.stream; }

    private DigitsStream(Stream<Integer> stream) {
        this.stream = stream;
    }

    public static DigitsStream of(Stream<Integer> stream) {
        return new DigitsStream(stream);
    }

    public int multiplyWith(int... factors) {
        return this.getStream().zip(this.repeatEndlessly(factors))
    }

    private Stream<Integer> repeatEndlessly(int... factors) {
        return Stream
            .iterate(factors, UnaryOperator.identity())
            .flatMapToInt(Arrays::stream)
            .boxed();
    }
}
```



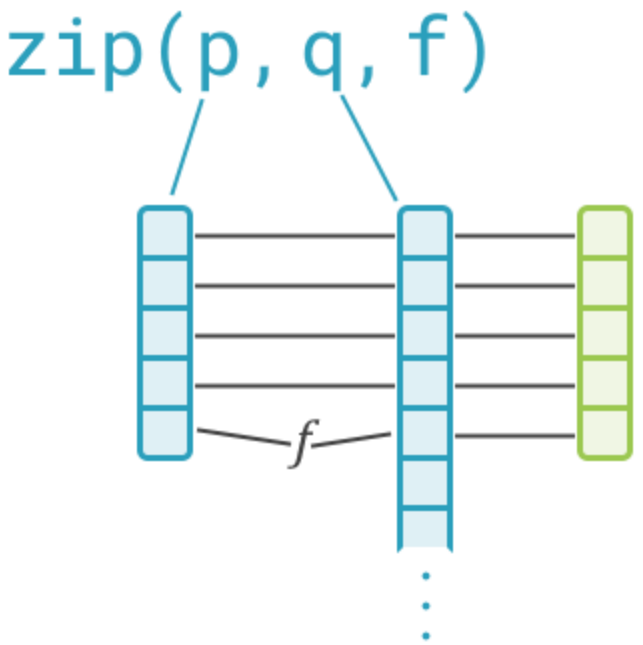
```java
public class DigitsStream implements ForwardingStream<Integer> {
    private Stream<Integer> stream;

    @Override
    public Stream<Integer> getStream() { return this.stream; }

    private DigitsStream(Stream<Integer> stream) {
        this.stream = stream;
    }

    public static DigitsStream of(Stream<Integer> stream) {
        return new DigitsStream(stream);
    }

    public int multiplyWith(int... factors) {
        return this.getStream().zip(this.repeatEndlessly(factors))
    }

    private Stream<Integer> repeatEndlessly(int... factors) {
        return Stream
            .iterate(factors, UnaryOperator.identity())
            .flatMapToInt(Arrays::stream)
            .boxed();
    }
}
```



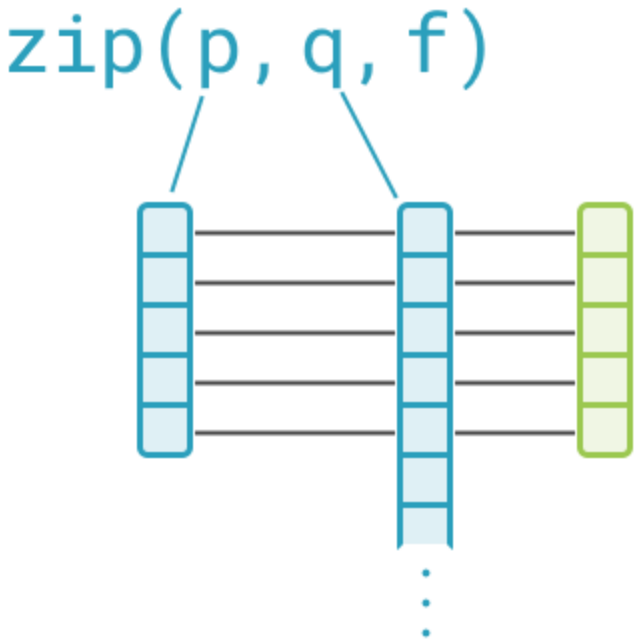
```java
public class DigitsStream implements ForwardingStream<Integer> {
    private Stream<Integer> stream;

    @Override
    public Stream<Integer> getStream() { return this.stream; }

    private DigitsStream(Stream<Integer> stream) {
        this.stream = stream;
    }

    public static DigitsStream of(Stream<Integer> stream) {
        return new DigitsStream(stream);
    }

    public int multiplyWith(int... factors) {
        return this.getStream().zip(this.repeatEndlessly(factors))
    }

    private Stream<Integer> repeatEndlessly(int... factors) {
        return Stream
            .iterate(factors, UnaryOperator.identity())
            .flatMapToInt(Arrays::stream)
            .boxed();
    }
}
```



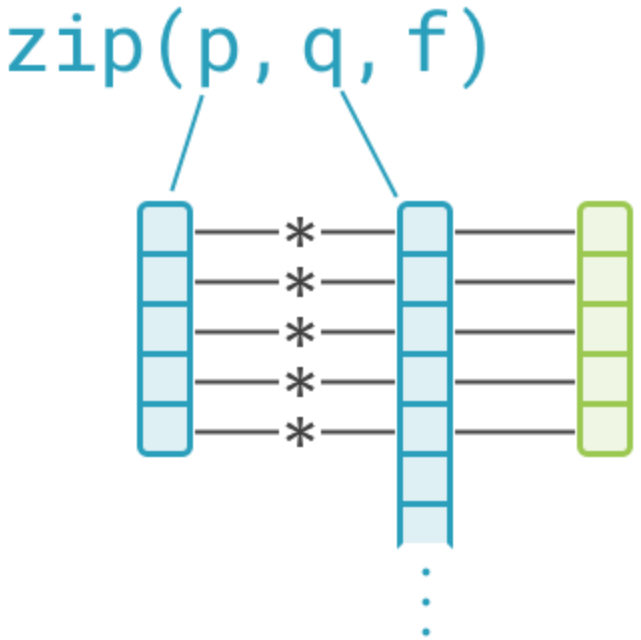
```java
public class DigitsStream implements ForwardingStream<Integer> {
    private Stream<Integer> stream;

    @Override
    public Stream<Integer> getStream() { return this.stream; }

    private DigitsStream(Stream<Integer> stream) {
        this.stream = stream;
    }

    public static DigitsStream of(Stream<Integer> stream) {
        return new DigitsStream(stream);
    }

    public int multiplyWith(int... factors) {
        return this.getStream().zip(this.repeatEndlessly(factors))
    }

    private Stream<Integer> repeatEndlessly(int... factors) {
        return Stream
            .iterate(factors, UnaryOperator.identity())
            .flatMapToInt(Arrays::stream)
            .boxed();
    }
}
```



```java
public class DigitsStream implements ForwardingStream<Integer> {
    private Stream<Integer> stream;

    @Override
    public Stream<Integer> getStream() { return this.stream; }

    private DigitsStream(Stream<Integer> stream) {
        this.stream = stream;
    }

    public static DigitsStream of(Stream<Integer> stream) {
        return new DigitsStream(stream);
    }

    public int multiplyWith(int... factors) {
        return this.getStream().zip(this.repeatEndlessly(factors))
    }

    private Stream<Integer> repeatEndlessly(int... factors) {
        return Stream
            .iterate(factors, UnaryOperator.identity())
            .flatMapToInt(Arrays::stream)
            .boxed();
    }
}
```



```java
public class DigitsStream implements ForwardingStream<Integer> {
    private Stream<Integer> stream;

    @Override
    public Stream<Integer> getStream() { return this.stream; }

    private DigitsStream(Stream<Integer> stream) {
        this.stream = stream;
    }

    public static DigitsStream of(Stream<Integer> stream) {
        return new DigitsStream(stream);
    }

    public int multiplyWith(int... factors) {
        return this.getStream().zip(this.repeatEndlessly(factors))
    }

    private Stream<Integer> repeatEndlessly(int... factors) {
        return Stream
            .iterate(factors, UnaryOperator.identity())
            .flatMapToInt(Arrays::stream)
            .boxed();
    }
}
```



```
public interface ControlDigit {
    static ControlDigitAlgorithm accountingAlgorithm() {
        return PonderingModuloAlgorithm.multipleDigitsModulo(
            StraightNumber::digitsFromLeastSignificant, divisor: 11, substitute: 0, ...factors: 3, 1);
    }

    static ControlDigitAlgorithm salesAlgorithmLegacy() {
        return PonderingModuloAlgorithm.singleDigitModulo(
            StraightNumber::digitsFromMostSignificant, divisor: 7, ...factors: 7, 3, 2);
    }
}
```

Hey mate!

We call it May 2017

Yup! We definitely call it May 2017

Hi guys!

What do you call that new algorithm?

What?!
No, that's a date

Summary

Implementing an object-oriented algorithm

- Separate infrastructural operations
- Focus on domain-related operations
- Wrap loops into objects (e.g. streams)
- Wrap transformations into meaningful methods

Summary

Exercising the algorithm's flexibility

- Easy to adapt to changing requirements
- Parameterize data and behavior
- Inject behavior as dependencies
- Give names to prominent sets of parameters
- Produce a flexible and maintainable algorithm implementation

Course Summary

Removing loops

- Turn their subjects into streams
- Stream encapsulates the loop
- Implement domain-specific streams
- Expose domain-specific methods
- Implement composite objects
- Composite behaves like a single object

Course Summary

Design a Domain-specific Language (DSL)

- Let complex transform read like a sentence
- Develop maintainable domain code

Course Summary

Removing multiway branching

- Organize rule objects into a graph
- Encapsulate selection and execution of domain logic
- Use rules to manage volatile business rules and flows

Course Summary

Turning algorithms into objects

- Make them maintainable in the long run

Course Summary

Enforcing the object-oriented method

- Improve coding skills
- Use design patterns
- Apply refactoring
- Incorporate functional design, because...
- THE BEST OBJECT-ORIENTED CODE IS FUNCTIONAL CODE