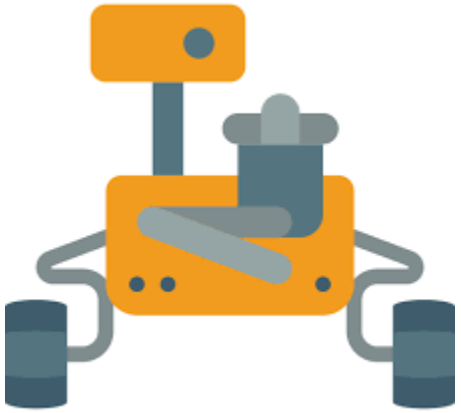
Mars rover exercise

Focus on coordinates inside Plateau check refactoring,
logic inside Coordinates class and Visitor pattern.
My personal training for study purposes.

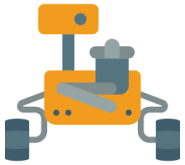


Giovanni Saba

giovanni.saba@adesso.ch

Agenda

- Problem of check coordinates inside Plateau
  - Code smell
- Refactoring approaches I attempted
  - Logic inside Coordinates class
  - Visitor pattern (attempt)
- Cohesion and coupling indices of the different approaches



Problem – code smell

```
public class Rover {

    public void execute(Command move) {
        if (move.equals(Command.MOVE)) {
            var tempPosition = position.deepClone();
            tempPosition.move();
            if(plateau.isValid(tempPosition.getCoordinates())) {
                position.move();
            }
        } else if (move.equals(Command.RIGHT)) {
            position.turnRight();
        }else if (move.equals(Command.LEFT)) {
            position.turnLeft();
        }
    }
...
}
```

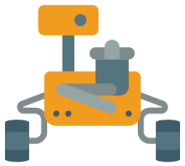
Long method

```
public class Plateau {
    private final int x;
    private final int y;

    ...

    public boolean isValid(Coordinates coordinates) {
        return coordinates.getX() >= 0 &&
               coordinates.getX() <= x &&
               coordinates.getY() >= 0 &&
               coordinates.getY() <= y;
    }
}
```

Primitive obsession
Data clumps

Feature envy

Mars rover exercise

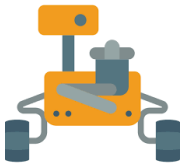


Split responsibilities in different methods

SRP

```
public class Rover {

    public void execute(Command command) {
        switch (command) {
            case MOVE -> {
                if (isMoveInsidePlateau()) {
                    position.move();
                }
            }
            case RIGHT -> position.turnRight();
            case LEFT -> position.turnLeft();
        }
    }

    private boolean isMoveInsidePlateau() {
        var attemptNextPosition = position.deepClone();
        attemptNextPosition.move();
        return plateau.coordinatesAreInsidePlateau(attemptNextPosition.getCoordinates());
    }
    …
}
```

Mars rover exercise

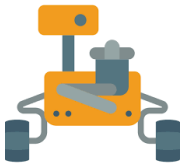


Let’s move logic inside Coordinates

```
public class Plateau {

   private final Coordinates upperRightCorner;
   private final Coordinates lowerLeftCoordinates;

   public Plateau(Coordinates coordinates) {
      this.upperRightCorner = coordinates;
      this.lowerLeftCoordinates = new Coordinates(0, 0);
   }

   public boolean coordinatesAreInsidePlateau(
               Coordinates coordinates) {
      return
                  !coordinates.isUpper(upperRightCorner)
            && !coordinates.isRight(upperRightCorner)
            && !coordinates.isLower(lowerLeftCoordinates)
            && !coordinates.isLeft(lowerLeftCoordinates);
   }
…
```

```
public class Coordinates {

   private int x;
   private int y;

   public Boolean isUpper(Coordinates referenceCoordinates) {
      return this.y > referenceCoordinates.y;
   }

   public Boolean isRight(Coordinates referenceCoordinates) {
      return this.x > referenceCoordinates.x;
   }

   public Boolean isLower(Coordinates referenceCoordinates) {
      return this.y < referenceCoordinates.y;
   }

   public Boolean isLeft(Coordinates referenceCoordinates) {
      return this.x < referenceCoordinates.x;
   }
```

Open Closed Principle



Cohesion and coupling

Class cohesion = $\frac{\sum_0^F n_{MF}}{n_M n_F}$

Class coupling = $1 - \frac{1}{d_I + d_O + g + k(cd_I + cd_O + cg) + m_I + m_O}$

nF ⇒number of fields
nM ⇒ number of methods
nMF ⇒ number of methods accessing the field F

dI ⇒ data in input
dO ⇒ data in output
cdI ⇒ control data input
cdO ⇒ control data output
g ⇒ global data
cg ⇒ control global data
mI ⇒ fan in (modules call in)
mO ⇒ fan out (modules call out)
K ⇒ control data crappiness constant

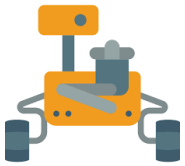
	Cohesion (Before)	Coupling (Before)	Cohesion (After)	Coupling (After)
Plateau	(1+1)/(2*2)=0.5	1 - 1 /(3 + 1 + 1(1+1) + 1 + 1) = 1 - 1/(8) = 0.875	(1+1)/(2*2)=0.5	1 - 1 /(1 + 1 + 1(1+1) + 1 + 1) = 1 - 1/(6) = 0.83
Coordinates	(1+1)/(1*2)=1	1 - 1 /(2 + 2 + 1(0+2) + 0 + 2) = 1 - 1/(8) = 0.875	(3+3)/(5*2)=0.6	1 - 1 /(2 + 4 + 1(1+4) + 1 + 2) = 1 - 1/(13) = 0.92
Rover	(1+1)/(1*2)=1	1 - 1 /(1 + 1 + 1(1) + 2 + 0) = 1 - 1/(5) = 0.8	(2+1)/(2*2)=0.75	1 - 1 /(1 + 1 + 1(1) + 2 + 0) = 1 - 1/(5) = 0.8

Outputs:

- No real improvements in cohesion and coupling (in some case even it gets worse, if my calculations are good)
- **But respect of SOLID principles like SRP and OCP**

* Calculations can be completely wrong (I used this scenario to train myself)

Mars rover exercise

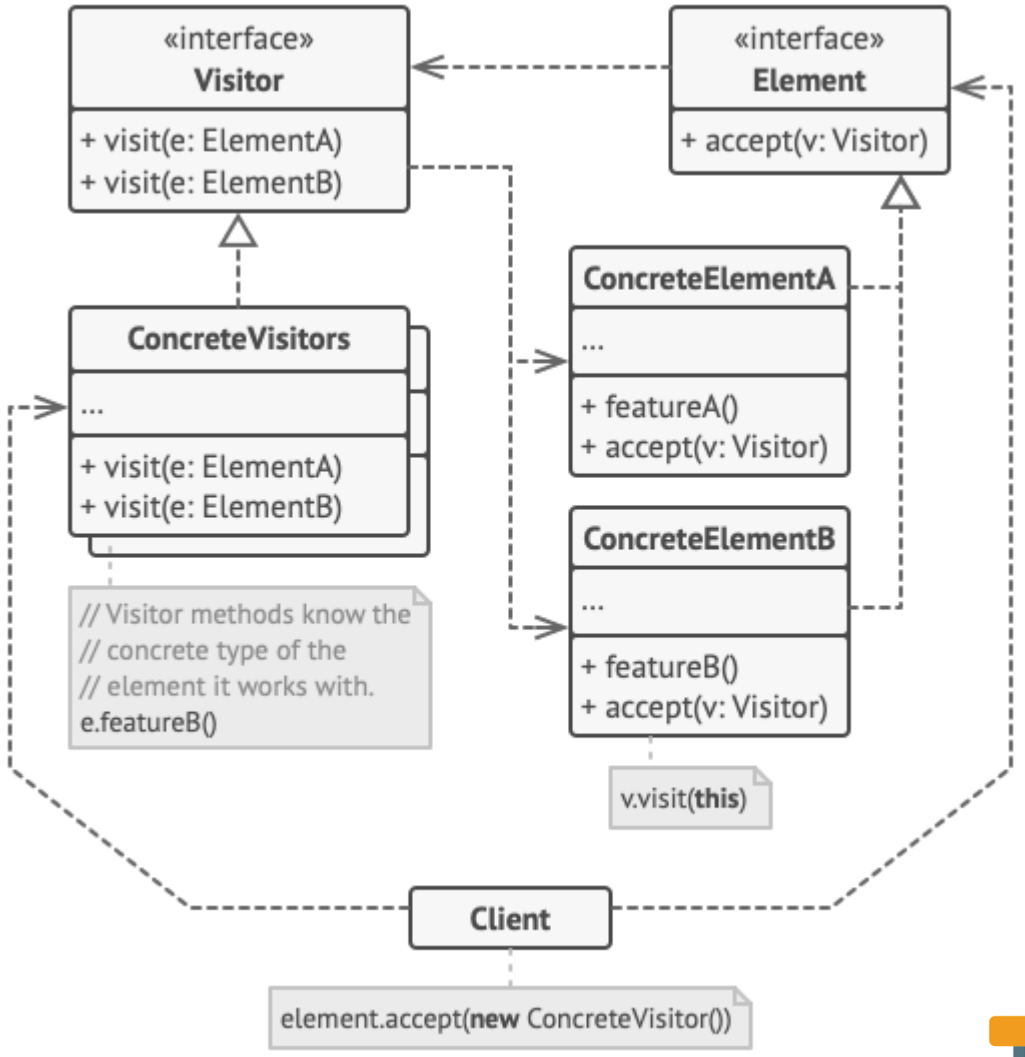


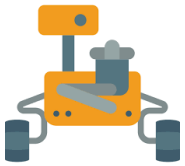
Visitor pattern

Assumption: **Not sure the way I implemented is the correct one, so let's check together…**

First of all what is the Visitor pattern?

Our need in this scenario is to compare 2 items, instead the Visitor suggests a way to access some properties of a class

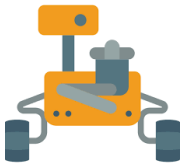




My Visitor pattern (is it correct?)

```
public class Plateau {

   private final Coordinates upperRightCorner;
   private final Coordinates lowerLeftCorner;

   public boolean isCoordinatesInPlateau(
            Coordinates coordinates) {
      var coordinatesComparator = new CoordinatesComparator();
      return
          !coordinates.isUpper(
              coordinatesComparator, upperRightCorner)
      && !coordinates.isRight(
              coordinatesComparator, upperRightCorner)
      && !coordinates.isLower(
              coordinatesComparator, lowerLeftCorner)
      && !coordinates.isLeft(
              coordinatesComparator, lowerLeftCorner);
   }
...
}
```

```
public class CoordinatesComparator {

   public boolean isUpper(Coordinates coordinates, Coordinates
referenceCoordinates) {
      return coordinates.getY() > referenceCoordinates.getY();
   }

   public boolean isRight(Coordinates coordinates, Coordinates
referenceCoordinates) {
      return coordinates.getX() > referenceCoordinates.getX();
   }

   public boolean isLower(Coordinates coordinates, Coordinates
referenceCoordinates) {
      return coordinates.getY() < referenceCoordinates.getY();
   }

   public boolean isLeft(Coordinates coordinates, Coordinates
referenceCoordinates) {
      return coordinates.getX() < referenceCoordinates.getX();
   }
}
...
```



Thanks

… any question?

References:

- Cohesion and coupling concepts and formulas: Alcor academy https://alcor.academy/
- Visitor pattern: https://refactoring.guru/design-patterns/visitor
- Source code: https://github.com/Alcor-Academy/eoc-ch-cohort-4/tree/experimental/gsaba