Group 22a Code Refactoring

---

1. Refactoring standards

The software we used for quantitative code quality metrics is the IntelliJ's MetricsTree plugin, mostly due to the ease of use and set-up, while also deemed as a sufficient solution for this assignment.

The thresholds we used for determining method/ class-level code smells are the same ones that the IntelliJ plugin MetricsTree suggests. For the metrics that do not have predefined appropriate ranges, we had to discuss with the group on a case-by-case basis depending on the context and how much difference the method/class gets relative to other methods/classes (so large outliers were generally prioritised for these metrics).

The metrics that we mostly focused on were cyclomatic complexity for method-level refactors. The solutions we mostly used were either to extract some functionality of a method, or rewriting the method to reduce CC.

For class-level refactors we looked at many more metrics rather than focusing on individual ones. More specifically, we were looking at the most widely used metrics to identify bloated, highly-coupled, and 'god' classes (a god class being a class that has too many responsibilities that can be seemingly unrelated). Some useful metrics for this include (but are not limited to) *Response for Class* (RFC), *Number of methods* (NOM), *Weighted methods per class* (WPC), *Coupling between Objects* (CBO), and *Access to Foreign Data (*ATFD).

All the code changes, along with the pertinent code metrics improvements will be shown back-to-back in the code snippets below, instead of laying them out here.

2. Actual refactoring

Method-level Refactors

1. HOA validator methods

We have identified that the HOA model validators have high Cyclomatic Complexity. More specifically, the Cyclomatic Complexity of *countryCheck is* 10. As indicated by the plugin, the average CC is below 3 so this required our attention.

```java
/**
 * This method checks that a country/city has the correct format.
 * This means that it must have maximum 50 characters, only letters (only ASCII), whitespaces are admissible,
 * no numbers and the first letter needs to be uppercase. Of course, the string cannot contain only whitespaces.
 *
 * @param country the country or city name
 * @return true if the country name fails to adhere to the format, false otherwise
 */
public boolean countryCheck(String country) {
    if (country == null || country.isEmpty() || country.isBlank()) {
        return true;
    }
    if (!Character.isUpperCase(country.charAt(0)) || country.length() < 4 || country.length() > 50) {
        return true;
    }
    for (int i = 1; i < country.length(); i++) {
        char c = country.charAt(i);
        if (!Character.isWhitespace(c) && !Character.isLetter(c)) {
            return true;
        }
    }
    return false;
}
```

CC	McCabe Cyclomatic Complexity	10	[0..3)

After applying proper refactoring techniques, the method has a CC of 2. The fix involved removing the unnecessary if statements and replacing them with a regex that mimics the previous behaviour.

```java
/**
 * This method checks that a country/city has the correct format.
 * This means that it must have maximum 50 characters, only letters (only ASCII), whitespaces are admissible,
 * no numbers and the first letter needs to be uppercase. Of course, the string cannot contain only whitespaces.
 *
 * @param country the country or city name
 * @return true if the country name fails to adhere to the format, false otherwise
 */
public boolean countryCheck(String country) {
    return country == null || !country.matches( regex: "^[A-Z][a-zA-Z\\s]{3,49}$");
}
```

CC	McCabe Cyclomatic Complexity	2	[0..3)

Another method with high CC is shown below. The method `nameCheck` utilised the helper method `enoughCharsAndWhitespace` in an attempt to reduce its own CC but it wasn't enough. Below you can see that both methods had high CC and a more elegant solution was required.

```java
/**
 * This is a method that checks whether the format for a name is correct.
 *
 * @param name the name of the hoa
 * @return true if the format is satisfied, false otherwise
 */
public boolean nameCheck(String name) {
    if (name == null || name.isEmpty() || name.isBlank()) {
        return false;
    }
    if (!Character.isUpperCase(name.charAt(0)) || name.length() > 50) {
        return false;
    }
    return enoughCharsAndWhitespace(name);
}

/**
 * Checks that a string has at least 4 characters.
 * Only letters or digits are allowed. The name can have at most 50 characters.
 *
 * @param name the name
 * @return true if the conditions are met, false otherwise.
 */
public boolean enoughCharsAndWhitespace(String name) {
    if (name == null || name.isEmpty() || name.isBlank()) {
        return false;
    }
    String[] split = name.split( regex: " ");
    StringBuilder sb = new StringBuilder();
    for (String s : split) {
        sb.append(s);
    }
    String result = sb.toString();
    char[] array = result.toCharArray();
    for (Character x : array) {
        if (!Character.isLetterOrDigit(x)) {
            return false;
        }
    }
    return result.length() >= 4;
}
```

Method: nameCheck(String)

Metric	Metrics Set	Description	Value	Regular Ra...
CND		Condition Nesting Depth	1	[0..2)
LND		Loop Nesting Depth	0	[0..2)
CC		McCabe Cyclomatic Complexity	6	[0..3)

Method: enoughCharsAndWhitespace(String)

Metric	Metrics Set	Description	Value	Regular Ra...
CND		Condition Nesting Depth	1	[0..2)
LND		Loop Nesting Depth	1	[0..2)
CC		McCabe Cyclomatic Complexity	7	[0..3)

The solution here was again a clever regex which removed the need of the if statements and of the helper method. In the end, from a total of 15 from both methods, we only have one method with a CC of 2.

```java
/**
 * Checks that a string has at least 4 characters.
 * Only letters or digits are allowed. The name can have at most 50 characters.
 *
 * @param name the name
 * @return true if the conditions are met, false otherwise.
 */
public boolean nameCheck(String name) {
    return name != null && name.matches( regex: "^(?!\\s*$)[a-zA-Z0-9\\s]{4,50}$");
}
```

Method: nameCheck(String)

Metric	Metrics Set	Description	Value	Regular Ra...
CND		Condition Nesting Depth	0	[0..2)
LND		Loop Nesting Depth	0	[0..2)
CC		McCabe Cyclomatic Complexity	2	[0..3)

2. Vote methods

Still using the MetricsTree IntelliJ plugin, we found out that these two methods below which are in the ElectionController class of the Voting microservice have high CC.

In this case it wasn't possible to avoid most of the if cases so we decided to extract parts of the methods into new ones. This is also a good decision here because some lines in the two methods were duplicated.

Complexity comparison:

Vote

Before:

Method: vote(VotingModel, LocalDateTime)

Metric	Metrics Set	Description	Value	Regular Ra...
CND		Condition Nesting Depth	1	[0..2)
LND		Loop Nesting Depth	0	[0..2)
CC		McCabe Cyclomatic Complexity	9	[0..3)

After:

Method: vote(VotingModel, LocalDateTime)

Metric	Metrics Set	Description	Value	Regula
CC		McCabe Cyclomatic Complexity	3	[0..3)

RemoveVote

Before:

Method: removeVote(RemoveVoteModel, LocalDateTime)

Metric	Metrics Set	Description	Value	Regular Ra...
CND		Condition Nesting Depth	1	[0..2)
LND		Loop Nesting Depth	0	[0..2)
CC		McCabe Cyclomatic Complexity	5	[0..3)

After:

Method: removeVote(RemoveVoteModel, LocalDateTime)

Metric	Metrics Set	Description	Value	Regular Ra...
CC		McCabe Cyclomatic Complexity	3	[0..3)

Code comparison:

```java
Method called when a member wants to vote
Params: model – VotingModel that contains electionId, memberID, and voting choice
Throws: ElectionDoesNotExist – If election does not exist with provided id
public void vote(VotingModel model, LocalDateTime currTime) throws ElectionDoesNotExist, CannotProceedVote {
    if (!model.isValid())
        throw new ElectionDoesNotExist("Ids not valid");
    Optional<Election> election = this.electionRepository.findByElectionId(model.electionId);
    if (election.isEmpty())
        throw new ElectionDoesNotExist("Election not found");
    if (election.get().getScheduledFor().isAfter(currTime))
        throw new CannotProceedVote("Election has not started");
    if (election.get().getStatus().equals("finished"))
        throw new CannotProceedVote("Election has been concluded");
    if (election.get().getClass() == BoardElection.class
            && !((BoardElection) election.get()).getCandidates().contains(model.choice))
        throw new CannotProceedVote("Candidate with given id is not nominated for the election");
    if (election.get().getClass() == Proposal.class
            && !List.of("True", "true", "T", "False", "false", "F").contains(model.choice))
        throw new CannotProceedVote("Invalid voting choice for proposal (must be a boolean or similar)");
    election.get().setStatus("ongoing");
    election.get().vote(model.memberId, model.choice);
    this.electionRepository.save(election.get());
}
```

```java
Method called when a member wants to remove his vote
Params: model – RemoveVoteModel that contains electionId and memberID
Throws: ElectionDoesNotExist – If election does not exist with provided id
        ThereIsNoVote – If the member has not voted yet
        CannotProceedVote – If the request has been made before the beginning or after the end of the
        voting process
public void removeVote(RemoveVoteModel model, LocalDateTime currTime)
                        throws ElectionDoesNotExist, ThereIsNoVote, CannotProceedVote {
    if (!model.isValid())
        throw new ElectionDoesNotExist("Ids not valid");
    Optional<Election> election = this.electionRepository.findByElectionId(model.electionId);
    if (election.isEmpty())
        throw new ElectionDoesNotExist("Election not found");
    if (election.get().getScheduledFor().isAfter(currTime))
        throw new CannotProceedVote("Election has not started");
    if (election.get().getStatus().equals("finished"))
        throw new CannotProceedVote("Election has been concluded");
    election.get().removeVote(model.memberId);
    this.electionRepository.save(election.get());
}
```

```java
public void vote(VotingModel model, LocalDateTime currTime) throws ElectionDoesNotExist, CannotProceedVote {
    if (!model.isValid())
        throw new ElectionDoesNotExist("Ids not valid");
    Optional<Election> election = this.electionRepository.findByElectionId(model.electionId);
    if (election.isEmpty())
        throw new ElectionDoesNotExist("Election not found");
    checkElectionTime(election.get(), currTime);
    checkElectionVote(election.get(), model.choice);
    election.get().setStatus("ongoing");
    election.get().vote(model.memberId, model.choice);
    this.electionRepository.save(election.get());
}

Method called when a member wants to remove his vote
Params: model – RemoveVoteModel that contains electionId and memberID
Throws: ElectionDoesNotExist – If election does not exist with provided id
        ThereIsNoVote – If the member has not voted yet
        CannotProceedVote – If the request has been made before the beginning or after the end of the
        voting process
public void removeVote(RemoveVoteModel model, LocalDateTime currTime)
                        throws ElectionDoesNotExist, ThereIsNoVote, CannotProceedVote {
    if (!model.isValid())
        throw new ElectionDoesNotExist("Ids not valid");
    Optional<Election> election = this.electionRepository.findByElectionId(model.electionId);
    if (election.isEmpty())
        throw new ElectionDoesNotExist("Election not found");
    checkElectionTime(election.get(), currTime);
    election.get().removeVote(model.memberId);
    this.electionRepository.save(election.get());
}
```

Added methods:

```java
Checks whether an election is ongoing in the current moment
Params: election – - the election for which the check is
        currTime – - the current moment
Throws: CannotProceedVote – - if the election is finished or not started
private void checkElectionTime(Election election, LocalDateTime currTime) throws CannotProceedVote {
    if (election.getScheduledFor().isAfter(currTime))
        throw new CannotProceedVote("Election has not started");
    if (election.getStatus().equals("finished"))
        throw new CannotProceedVote("Election has been concluded");
}

Checks whether the choice is a legal choice for the given election
Params: election – - the election that we check
        choice – - the choice that we check
Throws: CannotProceedVote – - if the vote is illegal
private void checkElectionVote(Election election, String choice) throws CannotProceedVote {

    if (election.getClass() == BoardElection.class) {
        if (!((BoardElection) election).getCandidates().contains(choice))
            throw new CannotProceedVote("Candidate with given id is not nominated for the election");
    } else if (!List.of("True", "true", "T", "False", "false", "F").contains(choice))
        throw new CannotProceedVote("Invalid voting choice for proposal (must be a boolean or similar)");
}
```

Method: checkElectionTime(Election, LocalDateTime)

Metric	Metrics Set	Description	Value	Regular Ra.
CC		McCabe Cyclomatic Complexity	3	[0..3)

Method: checkElectionVote(Election, String)

Metric	Metrics Set	Description	Value	Regular Ra.
CC		McCabe Cyclomatic Complexity	4	[0..3)

As you see on the pictures with the metrics the CC of all four of the methods now is at most 4 and the duplicated checks are extracted in new methods.

3. AuthMember validation methods

Similarly as in step one, AuthMember microservice also had 3 validation methods for membership, which were quite convoluted. We followed similar practices as before and used regex instead of a large number of IF statements.

Original methods:

```java
public boolean validatePostalCode(String postalCode) {
    if (postalCode == null) {
        return false;
    }
    String trimmed = postalCode.trim();
    final int max = 6;
    if (trimmed.length() != max) {
        return false;
    }
    for (int i = 0; i < 4; i++) {
        if (!Character.isDigit(trimmed.charAt(i))) {
            return false;
        }
    }
    for (int j = 4; j < 6; j++) {
        if (!Character.isLetter(trimmed.charAt(j))) {
            return false;
        }
    }
    return true;
}
```

```java
public boolean validateStreetNumber(String number) {
    if (number == null || number.isBlank() || number.isEmpty()) {
        return false;
    }
    String trimmed = number.trim();
    for (int i = 0; i < trimmed.length() - 1; i++) {
        if (!Character.isDigit(trimmed.charAt(i))) {
            return false;
        }
    }
    if (!Character.isDigit(trimmed.charAt(trimmed.length() - 1))
            && !Character.isLetter(trimmed.charAt(trimmed.length() - 1))) {
        return false;
    }
    return true;
}
```

```java
27 usages   Michal Krkoska *
public static boolean validateCountryCityStreet(String name) {
    if (checkNullBlankEmpty(name)) {
        return false;
    }
    String trimmed = name.trim();
    if (trimmed.length() < 4 || trimmed.length() > 50) {
        return false;
    }
    if (!Character.isUpperCase(trimmed.charAt(0))) {
        return false;
    }
    for (int i = 1; i < trimmed.length(); i++) {
        if (!Character.isLetter(trimmed.charAt(i)) && !Character.isWhitespace(trimmed.charAt(i)))
            return false;
        }
    }
    return true;

}
```

Original metrics:

validatePostalCode(String)
- Condition Nesting Depth: 1
- Halstead Difficulty: 17.7727
- Halstead Effort: 4613.7496
- Halstead Errors: 0.0924
- Halstead Length: 54
- Halstead Vocabulary: 28
- Halstead Volume: 259.5972
- Lines Of Code: 28
- Loop Nesting Depth: 1
- Maintainability Index: 51.2721
- McCabe Cyclomatic Complexity: 7
- Number Of Loops: 2
- Number Of Parameters: 1

validateStreetNumber(String)
- Condition Nesting Depth: 1
- Halstead Difficulty: 26.125
- Halstead Effort: 6832.179
- Halstead Errors: 0.12
- Halstead Length: 55
- Halstead Vocabulary: 27
- Halstead Volume: 261.5188
- Lines Of Code: 23
- Loop Nesting Depth: 1
- Maintainability Index: 53.0943
- McCabe Cyclomatic Complexity: 8
- Number Of Loops: 1
- Number Of Parameters: 1

validateCountryCityStreet(String)
- Condition Nesting Depth: 1
- Halstead Difficulty: 25.2
- Halstead Effort: 7365.8991
- Halstead Errors: 0.1262
- Halstead Length: 59
- Halstead Vocabulary: 31
- Halstead Volume: 292.2976
- Lines Of Code: 26
- Loop Nesting Depth: 1
- Maintainability Index: 51.5615
- McCabe Cyclomatic Complexity: 10
- Number Of Loops: 1
- Number Of Parameters: 1

Refactored methods:

```java
public static boolean validatePostalCode(String postalCode) {
    if (postalCode == null) {
        return false;
    }
    String trimmed = postalCode.trim();
    return trimmed.matches( regex: "[0-9]{4}[A-Za-z]{2}");
}
```

```java
public static boolean validateStreetNumber(String number) {
    if (checkNullBlankEmpty(number)) return false;
    String trimmed = number.trim();
    return trimmed.matches( regex: "^[0-9]*[A-Za-z]?$");
}
```

```java
public static boolean validateCountryCityStreet(String name) {
    if (checkNullBlankEmpty(name)) {
        return false;
    }
    String trimmed = name.trim();
    return trimmed.matches( regex: "[A-Z][A-Za-z]{3,49}");
}
```

Refactored methods metrics:

validatePostalCode(String)
- Condition Nesting Depth: 1
- Halstead Difficulty: 4.8
- Halstead Effort: 266.4317
- Halstead Errors: 0.0138
- Halstead Length: 15
- Halstead Vocabulary: 13
- Halstead Volume: 55.5066
- Lines Of Code: 14
- Loop Nesting Depth: 0
- Maintainability Index: 62.7191
- McCabe Cyclomatic Complexity: 2
- Number Of Loops: 0
- Number Of Parameters: 1

validateStreetNumber(String)
- Condition Nesting Depth: 1
- Halstead Difficulty: 5.0
- Halstead Effort: 250.9474
- Halstead Errors: 0.0133
- Halstead Length: 14
- Halstead Vocabulary: 12
- Halstead Volume: 50.1895
- Lines Of Code: 12
- Loop Nesting Depth: 0
- Maintainability Index: 64.4693
- McCabe Cyclomatic Complexity: 2
- Number Of Loops: 0
- Number Of Parameters: 1

validateCountryCityStreet(String)
- Condition Nesting Depth: 1
- Halstead Difficulty: 5.0
- Halstead Effort: 250.9474
- Halstead Errors: 0.0133
- Halstead Length: 14
- Halstead Vocabulary: 12
- Halstead Volume: 50.1895
- Lines Of Code: 15
- Loop Nesting Depth: 0
- Maintainability Index: 62.3553
- McCabe Cyclomatic Complexity: 2
- Number Of Loops: 0
- Number Of Parameters: 1

4. HoaService join method

Refactored the joinHoa method in Authservice by extracting logic to helper methods.

Original method:

```java
public String joinHoa(JoinHoaModel model, String token) throws
        MemberAlreadyInHoaException, MemberDifferentAddressException, BadJoinHoaModelException {
    try {
        List<Membership> activeMemberships = this.membershipService.getActiveMemberships(model.getMemberId());
        for (Membership membership : activeMemberships) {
            if (membership.getHoaId() == model.getHoaId()) {
                throw new MemberAlreadyInHoaException(model);
            }
        }
        HoaResponseModel hoa = HoaUtils.getHoaById(model.getHoaId(), token);
        if (!hoa.getCountry().equals(model.getAddress().getCountry())
                || !hoa.getCity().equals(model.getAddress().getCity())) {
            throw new MemberDifferentAddressException(model);
        }
        membershipService.saveMembership(model, asBoard: false);
        return model.getMemberId();
    } catch (IllegalArgumentException e) {
        throw new IllegalArgumentException("Hoa does not exist!");
    } catch (BadJoinHoaModelException e) {
        throw new BadJoinHoaModelException("Bad join hoa model!");
    }
}
```

Original metric:

```
joinHoa(JoinHoaModel, String)
    Condition Nesting Depth: 1
    Halstead Difficulty: 23.75
    Halstead Effort: 6943.7669
    Halstead Errors: 0.1213
    Halstead Length: 57
    Halstead Vocabulary: 35
    Halstead Volume: 292.3691
    Lines Of Code: 28
    Loop Nesting Depth: 1
    Maintainability Index: 50.9074
    McCabe Cyclomatic Complexity: 7
    Number Of Loops: 1
    Number Of Parameters: 2
```

Refactored method:

```java
public String joinHoa(JoinHoaModel model, String token) throws
        MemberAlreadyInHoaException, MemberDifferentAddressException, BadJoinHoaModelException {
    try {
        validateUserNotInHoa(model);
        HoaResponseModel hoa = HoaUtils.getHoaById(model.getHoaId(), token);
        validateUserInCorrectCity(model, hoa);
        membershipService.saveMembership(model, asBoard: false);
        return model.getMemberId();
    } catch (IllegalArgumentException e) {
        throw new IllegalArgumentException("Hoa does not exist!");
    } catch (BadJoinHoaModelException e) {
        throw new BadJoinHoaModelException("Bad join hoa model!");
    }
}
```

Refactored method metric:

joinHoa(JoinHoaModel, String)
- Condition Nesting Depth: 0
- Halstead Difficulty: 10.2143
- Halstead Effort: 1191.926
- Halstead Errors: 0.0375
- Halstead Length: 27
- Halstead Vocabulary: 20
- Halstead Volume: 116.6921
- Lines Of Code: 20
- Loop Nesting Depth: 0
- Maintainability Index: 57.0162
- McCabe Cyclomatic Complexity: 3
- Number Of Loops: 0
- Number Of Parameters: 2

Helper methods:

```java
public void validateUserNotInHoa(JoinHoaModel model) throws MemberAlreadyInHoaException {
    List<Membership> activeMemberships = this.membershipService.getActiveMemberships(model.getMemberId());
    for (Membership membership : activeMemberships) {
        if (membership.getHoaId() == model.getHoaId()) {
            throw new MemberAlreadyInHoaException(model);
        }
    }
}
```

```java
public void validateUserInCorrectCity(JoinHoaModel model, HoaResponseModel hoa) throws MemberDifferentAddressException {
    if (!hoa.getCountry().equals(model.getAddress().getCountry())
        || !hoa.getCity().equals(model.getAddress().getCity())) {
        throw new MemberDifferentAddressException(model);
    }
}
```

Helper methods metric:

```
validateUserInCorrectCity(JoinHoaModel, HoaResponseModel)
    Condition Nesting Depth: 1
    Halstead Difficulty: 17.5
    Halstead Effort: 1540.0
    Halstead Errors: 0.0445
    Halstead Length: 22
    Halstead Vocabulary: 16
    Halstead Volume: 88.0
    Lines Of Code: 11
    Loop Nesting Depth: 0
    Maintainability Index: 63.52
    McCabe Cyclomatic Complexity: 3
    Number Of Loops: 0
    Number Of Parameters: 2
```

```
validateUserNotInHoa(JoinHoaModel)
    Condition Nesting Depth: 1
    Halstead Difficulty: 9.75
    Halstead Effort: 757.2025
    Halstead Errors: 0.0277
    Halstead Length: 19
    Halstead Vocabulary: 17
    Halstead Volume: 77.6618
    Lines Of Code: 13
    Loop Nesting Depth: 1
    Maintainability Index: 62.3435
    McCabe Cyclomatic Complexity: 3
    Number Of Loops: 1
    Number Of Parameters: 1
```

5. MembershipSevice save method

Lastly, saveMembership method which combines all 3 methods mentioned above to validate the whole JoinHoaModel had a very high CC due to calling all the methods from within itself. Therefore just the validation functionality was extracted into a separate method called validate which resulted into two smaller and much more readable methods.

Original method:

```java
/**
 * Register a new membership.
 *
 * @throws MemberAlreadyInHoaException if there is an active membership for that HOA.
 */
public boolean saveMembership(JoinHoaModel model, boolean asBoard)
        throws MemberAlreadyInHoaException, BadJoinHoaModelException {
    if (!validateCountryCityStreet(model.getAddress().getCity())
            || !validateCountryCityStreet(model.getAddress().getCountry())
            || !validateCountryCityStreet(model.getAddress().getStreet())
            || !validateStreetNumber(model.getAddress().getHouseNumber())
            || !validatePostalCode(model.getAddress().getPostalCode())) {
        throw new BadJoinHoaModelException("Bad model!");
    }
    if (memberRepository.findByMemberId(model.getMemberId()).isEmpty()) {
        throw new IllegalArgumentException("Member not found!");
    }
    if (membershipRepository.findByMemberIdAndHoaIdAndDurationIsNull(model.getMemberId(),
            model.getHoaId()).isPresent()) {
        throw new MemberAlreadyInHoaException(model);
    } else {
        membershipRepository.save(new Membership(model.getMemberId(),
                model.getHoaId(), model.getAddress(), LocalDateTime.now(), duration: null, asBoard));
        return true;
    }
}
```

Original metric:

```
saveMembership(JoinHoaModel, boolean)
    Condition Nesting Depth: 1
    Halstead Difficulty: 32.5
    Halstead Effort: 11243.2929
    Halstead Errors: 0.1673
    Halstead Length: 68
    Halstead Vocabulary: 34
    Halstead Volume: 345.9475
    Lines Of Code: 26
    Loop Nesting Depth: 0
    Maintainability Index: 51.0843
    McCabe Cyclomatic Complexity: 8
    Number Of Loops: 0
    Number Of Parameters: 2
```

Refactored method:

```java
/**
 * Register a new membership.
 *
 * @throws MemberAlreadyInHoaException if there is an active membership for that HOA.
 */
public boolean saveMembership(JoinHoaModel model, boolean asBoard)
        throws MemberAlreadyInHoaException, BadJoinHoaModelException {
    if (!validate(model)) {
        throw new BadJoinHoaModelException("Bad model!");
    }
    if (memberRepository.findByMemberId(model.getMemberId()).isEmpty()) {
        throw new IllegalArgumentException("Member not found!");
    }
    if (membershipRepository.findByMemberIdAndHoaIdAndDurationIsNull(model.getMemberId(),
            model.getHoaId()).isPresent()) {
        throw new MemberAlreadyInHoaException(model);
    } else {
        membershipRepository.save(new Membership(model.getMemberId(),
                model.getHoaId(), model.getAddress(), LocalDateTime.now(), duration: null, asBoard));
        return true;
    }
}
```

Metric after:

```
saveMembership(JoinHoaModel, boolean)
    Condition Nesting Depth: 1
    Halstead Difficulty: 19.0
    Halstead Effort: 4155.7717
    Halstead Errors: 0.0862
    Halstead Length: 46
    Halstead Vocabulary: 27
    Halstead Volume: 218.7248
    Lines Of Code: 22
    Loop Nesting Depth: 0
    Maintainability Index: 54.1561
    McCabe Cyclomatic Complexity: 4
    Number Of Loops: 0
    Number Of Parameters: 2
```

Extracted method:

```java
/**
 * Validates the input for JoinHoaModel
 *
 * @param model JoinHoaModel model to validate
 * @return true if the name satisfies the right format
 */
public static boolean validate(JoinHoaModel model) {
    return validateCountryCityAndStreet(model)
            && validateStreetNumber(model.getAddress().getHouseNumber())
            && validatePostalCode(model.getAddress().getPostalCode());
}
```

Extracted method metric:

```
validate(JoinHoaModel)
    Condition Nesting Depth: 0
    Halstead Difficulty: 15.0
    Halstead Effort: 726.4806
    Halstead Errors: 0.0269
    Halstead Length: 14
    Halstead Vocabulary: 11
    Halstead Volume: 48.432
    Lines Of Code: 11
    Loop Nesting Depth: 0
    Maintainability Index: 65.3633
    McCabe Cyclomatic Complexity: 3
    Number Of Loops: 0
    Number Of Parameters: 1
```

Class-level Refactors

Voting MS Models (6 classes)

The way election models were previously implemented went against some common programming principles for data carriers and models. For one, all the models used the @Data annotation from Lombok, even though all the fields were public, counteracting the usefulness of getters and setters. Furthemore, the Data objects were treated as regular entities (in which an equals method will go through all fields) and the fields were mutable in nature. However, data-carriers and models are usually immutable by nature, hence every model is unique. Finally, the models were lacking in their constructor design, inhibiting many shortcuts. For instance, the ProposalModel previously had to be instantiated with a defaulted no arg constructor, and then set the fields one by one. All of this led to bloated model classes that had high responsibility, high WMC (weighted methods/class) and a lot of overloaded/superfluous methods.

Hence, we refactored all the models in the voting/election microservice and transformed them into immutable, easily constructable DTO models, as they were intended to be in the first place. This reduced the number of methods (both overloaded and regular), as we only leave the most important methods instead of letting Lombok generate all the boilerplate for us, alleviated some of the RFC, and made the models generally more usable. A top-to-bottom comparison of each model, along with the most important affected metrics can be seen below (before/after refactoring on the left/right respectively for the metrics comparison).

1.ProposalModel

```
29 usages   Atanas Semov +1
@Data
@EqualsAndHashCode(callSuper = true)
public class ProposalModel extends ElectionModel {

}
```

```
31 usages   Atanas Semov *
public class ProposalModel extends ElectionModel {

    /** Create an ElectionModel object from base fields ...*/
    8 usages   new *
    public ProposalModel(String name, String description, long hoaId, TimeModel scheduledFor) {
        super(name, description, hoaId, scheduledFor);
    }
}
```

Class: ProposalModel

M...	Metrics Set	Description	Value	Regular Ra.
NOAC	Lanza-Marin...	Number Of Accessor Methods	0	[0..4)
NOAM	Lorenz-Kidd ...	Number Of Added Methods	0	
NOM	Li-Henry Met...	Number Of Methods	5	[0..7)
NOO	Lorenz-Kidd ...	Number Of Operations	32	
NOOM	Lorenz-Kidd ...	Number Of Overridden Methods	4	[0..3)
RFC	Chidamber-K...	Response For A Class	5	[0..45)
WMC	Chidamber-K...	Weighted Methods Per Class	9	[0..12)

Class: ProposalModel

M...	Metrics Set	Description	Value	Regular Ra.
NOAC	Lanza-Mari...	Number Of Accessor Methods	0	[0..4)
NOAM	Lorenz-Kid...	Number Of Added Methods	0	
NOM	Li-Henry M...	Number Of Methods	1	[0..7)
NOO	Lorenz-Kid...	Number Of Operations	19	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
RFC	Chidamber-...	Response For A Class	2	[0..45)
WMC	Chidamber-...	Weighted Methods Per Class	1	[0..12)

2.BoardElectionModel

```
24 usages   Alperen Guncan +2
@Data
@EqualsAndHashCode(callSuper = true)
public class BoardElectionModel extends ElectionModel {
    13 usages
    public int amountOfWinners;
    public List<String> candidates;

    /**
     * {@inheritDoc}
     */
    Alperen Guncan
    @Override
    public boolean isValid() {
        return amountOfWinners > 0
                && candidates != null
                && super.isValid();
    }
}
```

```java
29 usages   Alperen Guncan +1 *
public class BoardElectionModel extends ElectionModel {
    11 usages
    public final int amountOfWinners;
    public final List<String> candidates;

    /** Default constructors for [de-]serializer */
    no usages   new *
    public BoardElectionModel() {
        amountOfWinners = 0;
        candidates = null;
    }

    /** Creates a board election model from supermodel ...*/
    8 usages   new *
    public BoardElectionModel(ElectionModel em, int amountOfWinners, List<String> candidates) {
        this(em.name, em.description, em.hoaId, em.scheduledFor, amountOfWinners, candidates);
    }

    /** Creates a board election model from base fields ...*/
    4 usages   new *
    public BoardElectionModel(String name, String description, long hoaId, TimeModel scheduledFor,
                              int amountOfWinners, List<String> candidates) {
        super(name, description, hoaId, scheduledFor);
        this.amountOfWinners = amountOfWinners;
        this.candidates = candidates;
    }
```

```java
/** {@inheritDoc} */
Alperen Guncan
@Override
public boolean isValid() {
    return amountOfWinners > 0
            && candidates != null
            && super.isValid();
}
```

Class: BoardElectionModel

M...	Metrics Set	Description	Value	Regular Ra..
NOAC	Lanza-Marin...	Number Of Accessor Methods	2	[0..4)
NOAM	Lorenz-Kidd ...	Number Of Added Methods	4	
NOM	Li-Henry Met...	Number Of Methods	10	[0..7)
NOO	Lorenz-Kidd ...	Number Of Operations	37	
NOOM	Lorenz-Kidd ...	Number Of Overridden Methods	5	[0..3)
RFC	Chidamber-K...	Response For A Class	11	[0..45)
WMC	Chidamber-K...	Weighted Methods Per Class	20	[0..12)

Class: BoardElectionModel

M...	Metrics Set	Description	Value	Regular Ra..
NOAC	Lanza-Mari...	Number Of Accessor Methods	0	[0..4)
NOAM	Lorenz-Kid...	Number Of Added Methods	0	
NOM	Li-Henry M...	Number Of Methods	4	[0..7)
NOO	Lorenz-Kid...	Number Of Operations	22	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	1	[0..3)
RFC	Chidamber-...	Response For A Class	6	[0..45)
WMC	Chidamber-...	Weighted Methods Per Class	6	[0..12)

3.ElectionModel

```
9 usages   2 inheritors   Alperen Guncan +2
@Data
public class ElectionModel {
    public long hoaId;
    public String name;
    public String description;
    /** For board Election this is the scheduled time
    public TimeModel scheduledFor;

    /** Overwrite base DTO constructor to change visib
    2 usages   Alperen Guncan
    protected ElectionModel() {
        // This constructor cannot be called, hence it
    }

    /** Checks whether this model is a valid one for c
    1 override   Alperen Guncan
    public boolean isValid() {
        return hoaId > 0
                && name != null
                && name.length() > 0
                && name.length() < 90
                && description != null
                && description.length() > 0
                && description.length() < 240
                && scheduledFor != null
                && scheduledFor.isValid();
    }
```

```
2 inheritors   Alperen Guncan +1 *
public class ElectionModel {

    public final long hoaId;
    public final String name;
    public final String description;
    /** For board Election this is the scheduled time for voting ...*/
    public final TimeModel scheduledFor;

    /** Default constructors for [de-]serializer */
    new *
    public ElectionModel() {
        hoaId = 0;
        name = "";
        description = "";
        scheduledFor = null;
    }
    /** Create an ElectionModel object from base fields ...*/
    new *
    public ElectionModel(String name, String description, long hoaId, TimeModel scheduledFor) {
        this.name = name;
        this.description = description;
        this.hoaId = hoaId;
        this.scheduledFor = scheduledFor;
    }
```

```java
/** Create an ElectionModel object from base fields ...*/
Alperen Guncan *
public ElectionModel(String name, String description, long hoaId, LocalDateTime sf) {
    this(name, description, hoaId, new TimeModel(sf.getSecond(), sf.getMinute(),
            sf.getHour(), sf.getDayOfMonth(), sf.getMonthValue(), sf.getYear()));
}

/** Checks whether this model is a valid one for creating an election ...*/
1 override  Alperen Guncan
public boolean isValid() {
    return hoaId > 0
            && name != null
            && name.length() > 0
            && name.length() < 90
            && description != null
            && description.length() > 0
            && description.length() < 240
            && scheduledFor != null
            && scheduledFor.isValid();
}

new *
@Override
public boolean equals(Object o) {
    return this.getClass() == o.getClass() && this == o;
}
```

Class: ElectionModel

M...	Metrics Set	Description	Value	Regular Ra..
NOAC	Lanza-Marin...	Number Of Accessor Methods	4	[0..4)
NOAM	Lorenz-Kidd ...	Number Of Added Methods	10	
NOM	Li-Henry Met...	Number Of Methods	14	[0..7)
NOO	Lorenz-Kidd ...	Number Of Operations	27	
NOOM	Lorenz-Kidd ...	Number Of Overridden Methods	3	[0..3)
RFC	Chidamber-K...	Response For A Class	16	[0..45)
WMC	Chidamber-K...	Weighted Methods Per Class	35	[0..12)

Class: ElectionModel

M...	Metrics Set	Description	Value	Regular Ra.
NOAC	Lanza-Mari...	Number Of Accessor Methods	0	[0..4)
NOAM	Lorenz-Kid...	Number Of Added Methods	1	
NOM	Li-Henry M...	Number Of Methods	5	[0..7)
NOO	Lorenz-Kid...	Number Of Operations	18	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	1	[0..3)
RFC	Chidamber-...	Response For A Class	15	[0..45)
WMC	Chidamber-...	Weighted Methods Per Class	14	[0..12)

4.RemoveVoteModel

```java
@Data
@AllArgsConstructor
public class RemoveVoteModel {

    public int electionId;
    1 usage
    public String memberId;

    /**
     * Checks whether this model is a
     *
     * @return Boolean to represent the
     */
    Atanas Semov
    public boolean isValid() {
        return electionId >= 0;
    }
}
```

```java
@AllArgsConstructor
public class RemoveVoteModel {

    public final int electionId;
    2 usages
    public final String memberId;

    /** Checks whether this model is a valid one for removing a vote ...*/
    Atanas Semov
    public boolean isValid() {
        return electionId >= 0;
    }

    new *
    @Override
    public boolean equals(Object o) {
        return this.getClass() == o.getClass() && this == o;
    }
}
```

Class: RemoveVoteModel

M...	Metrics Set	Description	Value	Regular Ra.
NOAC	Lanza-Marin...	Number Of Accessor Methods	2	[0..4)
NOAM	Lorenz-Kidd ...	Number Of Added Methods	6	
NOM	Li-Henry Met...	Number Of Methods	10	[0..7)
NOO	Lorenz-Kidd ...	Number Of Operations	23	
NOOM	Lorenz-Kidd ...	Number Of Overridden Methods	3	[0..3)
RFC	Chidamber-K...	Response For A Class	10	[0..45)
WMC	Chidamber-K...	Weighted Methods Per Class	17	[0..12)

Class: RemoveVoteModel

M...	Metrics Set	Description	Value	Regular Ra.
NOAC	Lanza-Mari...	Number Of Accessor Methods	0	[0..4)
NOAM	Lorenz-Kid...	Number Of Added Methods	1	
NOM	Li-Henry M...	Number Of Methods	3	[0..7)
NOO	Lorenz-Kid...	Number Of Operations	16	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	1	[0..3)
RFC	Chidamber-...	Response For A Class	4	[0..45)
WMC	Chidamber-...	Weighted Methods Per Class	4	[0..12)

5.VotingModel

```java
@Data
@AllArgsConstructor
public class VotingModel {
    public int electionId;
    6 usages
    public String memberId;
    5 usages
    public String choice;

    /**
     * Checks whether this model is a valid one
     *
     * @return Boolean to represent the model's
     */
    Alperen Guncan
    public boolean isValid() {
        return electionId >= 0
                && !choice.equals(memberId);
    }
}
```

```java
Alperen Guncan +1 *
@AllArgsConstructor
public class VotingModel {
    public final int electionId;
    7 usages
    public final String memberId;
    6 usages
    public final String choice;

    /** Checks whether this model is a valid one for creating a vote ...*/
    Alperen Guncan
    public boolean isValid() {
        return electionId >= 0
                && !choice.equals(memberId);
    }

    new *
    @Override
    public boolean equals(Object o) {
        return this.getClass() == o.getClass() && this == o;
    }
}
```

Class: VotingModel

M...	Metrics Set	Description	Value	Regular Ra.
NOAC	Lanza-Marin...	Number Of Accessor Methods	3	[0..4)
NOAM	Lorenz-Kidd ...	Number Of Added Methods	8	
NOM	Li-Henry Met...	Number Of Methods	12	[0..7)
NOO	Lorenz-Kidd ...	Number Of Operations	25	
NOOM	Lorenz-Kidd ...	Number Of Overridden Methods	3	[0..3)
RFC	Chidamber-K...	Response For A Class	13	[0..45)
WMC	Chidamber-K...	Weighted Methods Per Class	23	[0..12)

Class: VotingModel

M...	Metrics Set	Description	Value	Regular Ra..
NOAC	Lanza-Mari...	Number Of Accessor Methods	0	[0..4)
NOAM	Lorenz-Kid...	Number Of Added Methods	1	
NOM	Li-Henry M...	Number Of Methods	3	[0..7)
NOO	Lorenz-Kid...	Number Of Operations	16	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	1	[0..3)
RFC	Chidamber-...	Response For A Class	5	[0..45)
WMC	Chidamber-...	Weighted Methods Per Class	5	[0..12)

6.TimeModel

```java
Alperen Guncan +1
@Data
@AllArgsConstructor
public class TimeModel {
    1 usage
    public int seconds;
    1 usage
    public int minutes;
    1 usage
    public int hours;
    3 usages
    public int day;
    1 usage
    public int month;
    2 usages
    public int year;

    Alperen Guncan
    public boolean isValid() {
        return createDate() != null;
    }

    /** Returns a new LocalDateTime object based on this model's attributes ...*/
    29 usages  Alperen Guncan
    public LocalDateTime createDate() {
        try {
            return LocalDateTime.of(LocalDate.of(this.year, this.month, this.day),
                    LocalTime.of(this.hours, this.minutes, this.seconds));
        } catch (DateTimeException e) {
            return null;
        }
    }
}
```

```java
Alperen Guncan +1 *
@AllArgsConstructor
public class TimeModel {
    4 usages
    public final int seconds;
    4 usages
    public final int minutes;
    4 usages
    public final int hours;
    2 usages
    public final int day;
    4 usages
    public final int month;
    4 usages
    public final int year;

    Alperen Guncan
    public boolean isValid() {
        return createDate() != null;
    }

    /** Returns a new LocalDateTime object based on this model's attributes ...*/
    30 usages  Alperen Guncan
    public LocalDateTime createDate() {
        try {
            return LocalDateTime.of(LocalDate.of(this.year, this.month, this.day),
                    LocalTime.of(this.hours, this.minutes, this.seconds));
        } catch (DateTimeException e) {
            return null;
        }
    }

    new *
    @Override
    public boolean equals(Object o) {
        return this.getClass() == o.getClass() && this == o;
    }
}
```

Class: TimeModel

M...	Metrics Set	Description	Value	Regular Ra.
NOAC	Lanza-Marin...	Number Of Accessor Methods	6	[0..4)
NOAM	Lorenz-Kidd ...	Number Of Added Methods	15	
NOM	Li-Henry Met...	Number Of Methods	19	[0..7)
NOO	Lorenz-Kidd ...	Number Of Operations	32	
NOOM	Lorenz-Kidd ...	Number Of Overridden Methods	3	[0..3)
RFC	Chidamber-K...	Response For A Class	22	[0..45)
WMC	Chidamber-K...	Weighted Methods Per Class	29	[0..12)

Class: TimeModel

M...	Metrics Set	Description	Value	Regular Ra...
NOAC	Lanza-Mari...	Number Of Accessor Methods	0	[0..4)
NOAM	Lorenz-Kid...	Number Of Added Methods	2	
NOM	Li-Henry M...	Number Of Methods	4	[0..7)
NOO	Lorenz-Kid...	Number Of Operations	17	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	1	[0..3)
RFC	Chidamber-...	Response For A Class	8	[0..45)
WMC	Chidamber-...	Weighted Methods Per Class	6	[0..12)

Refactored JoinHoaModel and MembershipResponseModel by removing unnecessary Lombok annotations and either changing them to more restrictive ones or writing them by hand.

7.Refactoring HOAController

Analysing the classes with MetricsTree, we noticed that the HOAController class had quite a high CBO value, and a high number of methods. Inspecting the code present in the class, a fact became quite clear, HOAController was also holding the responsibility of handling requirements. It only made sense to extract a class that we named RequirementController, to delegate some of the responsibility of the initial class. The CK metrics evidently improved afterwards.

Metrics before:

Class: HoaController

Metric	Value	Excess
Access To Foreign Data	6	+1
Coupling Between Objects	14	+1
Data Abstraction Coupling	3	-
Depth Of Inheritance Tree	1	
Halstead Difficulty	53.1...	-
Halstead Effort	8434...	-
Halstead Errors	0.6411	-
Halstead Length	240	-
Halstead Vocabulary	98	-
Halstead Volume	1587....	-
Lack Of Cohesion Of Methods	1	-
Maintainability Index	28.2...	+9.2...
Message Passing Coupling	52	-
Non-Commenting Source Statements	48	
Number Of Accessor Methods	2	
Number Of Added Methods	10	-
Number Of Attributes	3	
Number Of Attributes And Methods	26	-
Number Of Children	0	
Number Of Methods	11	+5
Number Of Operations	23	-
Number Of Overridden Methods	0	
Number Of Public Attributes	0	
Response For A Class	50	+6
Tight Class Cohesion	0.6889	
Weight Of A Class	0.8	
Weighted Methods Per Class	25	+14

Metrics after:

Class: HoaController

Metric	Value	Excess
Access To Foreign Data	5	
Coupling Between Objects	9	
Data Abstraction Coupling	1	-
Depth Of Inheritance Tree	1	
Halstead Difficulty	33.8...	-
Halstead Effort	2794...	-
Halstead Errors	0.3069	-
Halstead Length	132	-
Halstead Vocabulary	76	-
Halstead Volume	824....	-
Lack Of Cohesion Of Methods	1	-
Maintainability Index	36.2...	+17.2...
Message Passing Coupling	31	-
Non-Commenting Source Statements	24	
Number Of Accessor Methods	1	
Number Of Added Methods	5	-
Number Of Attributes	1	
Number Of Attributes And Methods	19	-
Number Of Children	0	
Number Of Methods	6	
Number Of Operations	18	-
Number Of Overridden Methods	0	
Number Of Public Attributes	0	
Response For A Class	35	
Tight Class Cohesion	1.0	
Weight Of A Class	0.8	
Weighted Methods Per Class	14	+3

Class: RequirementController

Metric	Value	Excess
Access To Foreign Data	4	
Coupling Between Objects	8	
Data Abstraction Coupling	3	-
Depth Of Inheritance Tree	1	
Halstead Difficulty	26.6...	-
Halstead Effort	1851...	-
Halstead Errors	0.2333	-
Halstead Length	118	-
Halstead Vocabulary	59	-
Halstead Volume	694....	-
Lack Of Cohesion Of Methods	1	-
Maintainability Index	37.0...	+18....
Message Passing Coupling	21	-
Non-Commenting Source Statements	25	
Number Of Accessor Methods	1	
Number Of Added Methods	5	-
Number Of Attributes	3	
Number Of Attributes And Methods	21	-
Number Of Children	0	
Number Of Methods	6	
Number Of Operations	18	-
Number Of Overridden Methods	0	
Number Of Public Attributes	0	
Response For A Class	27	
Tight Class Cohesion	1.0	
Weight Of A Class	0.8	
Weighted Methods Per Class	12	+1

The code before refactoring:
HoaController

```java
package nl.tudelft.sem.template.hoa.controllers;

import ...

/**
 * The controller for the association.
 */
1 usage  Alperen Guncan +3
@RestController
@RequestMapping("/hoa")
public class HoaController {

    7 usages
    private transient HoaService hoaService;
    6 usages
    private transient RequirementService requirementService;
    2 usages
    private final transient HoaRepo hoaRepo;


    /** Constructor for the HoaController. ...*/
    no usages  Rares Toader +1
    @Autowired
    public HoaController(HoaService hoaService, RequirementService requirementService,
                         HoaRepo hoaRepo) {
        this.hoaService = hoaService;
        this.requirementService = requirementService;
        this.hoaRepo = hoaRepo;
    }

    /** Endpoint for creating an association. ...*/
    no usages  Alperen Guncan +1
    @PostMapping(value = {"/create", "/create/{scheduledAfterXDays}"})
    public ResponseEntity<Hoa> register(@RequestBody HoaRequestModel request,
                                        @PathVariable Optional<Integer> scheduledAfterXDays) {
        try {
            Hoa newHoa = this.hoaService.registerHoa(request);
            if (scheduledAfterXDays.isEmpty() || scheduledAfterXDays.get() < 0) return ResponseEntity.ok(newHoa);
            Integer[] nums = Arrays.stream(LocalDateTime.now().plusDays(scheduledAfterXDays.get()) LocalDateTime
                    .format(DateTimeFormatter.ISO_DATE_TIME) String
                    .split( regex: "\\D+")).map(Integer::parseInt).toArray(Integer[]::new);
            // start automatic annual board election
            Object e = ElectionUtils.createBoardElection(new BoardElectionRequestModel(newHoa.getId(),
                    amountOfWinners: 2, List.of(), name: "Annual board election",
                    description: "This is the auto-generated annual board election",
                    TimeModel.createModelFromArr(nums)));
            if (e != null) return ResponseEntity.ok(newHoa);
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Could not create a board election");
        } catch (Exception e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, e.getMessage(), e);
        }
    }


    /** Endpoint for getting all HOAs in the database. ...*/
    no usages  Rares Toader +2
    @GetMapping("/getAll")
    public ResponseEntity<List<Hoa>> getAll() {
        try {
            return ResponseEntity.ok(hoaService.getAllHoa());
        } catch (Exception e) {
            return ResponseEntity.badRequest().build();
        }
    }

    /** Endpoint for getting one HOA from the database. ...*/
    no usages  Rares Toader +2
    @GetMapping("/getById/{id}")
    public ResponseEntity<Hoa> getById(@PathVariable long id) {
        try {
            return ResponseEntity.ok(hoaService.getHoaById(id));
        } catch (Exception e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Could not fetch HOA", e);
        }
    }

    /** Gets all requirements of an HOA ...*/
    no usages  Alperen Guncan
    @GetMapping("/getRequirements/{hoaId}")
    public ResponseEntity<List<Requirement>> getRequirements(@PathVariable long hoaId) {
        try {
            if (hoaRepo.findById(hoaId).isEmpty())
                throw new HoaDoesntExistException("Hoa with provided id does not exist");
            return ResponseEntity.ok(requirementService.getHoaRequirements(hoaId));
        } catch (Exception e) {
            return ResponseEntity.badRequest().build();
        }
    }

    /** Adds a requirement to an HOA ...*/
    no usages  Alperen Guncan +1
    @PostMapping("/addRequirement/{hoaId}")
    public ResponseEntity<Requirement> addRequirement(@PathVariable long hoaId,
                                                      @RequestBody Object prompt) {
        try {
            Requirement req = requirementService.addHoaRequirement(hoaId, String.valueOf(prompt));
            return ResponseEntity.ok(req);
        } catch (RequirementAlreadyPresent | HoaDoesntExistException e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, e.getMessage(), e);
        }
    }

    /** Removes a requirement from an HOA ...*/
    no usages  Alperen Guncan +1
    @PostMapping("/removeRequirement/{reqId}")
    public ResponseEntity<Requirement> removeRequirements(@PathVariable long reqId) {
        try {
            Requirement req = requirementService.removeHoaRequirement(reqId);
            return ResponseEntity.ok(req);
        } catch (RequirementDoesNotExist e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, e.getMessage(), e);
        }
    }

    /** Setter method used when HoaService needs to be mocked ...*/
    1 usage  Michal Krkoska
    public void setHoaService(HoaService h) { this.hoaService = h; }

    /** Setter method used when RequirementService needs to be mocked ...*/
    6 usages  Michal Krkoska
    public void setRequirementService(RequirementService h) { this.requirementService = h; }

    /** Endpoint for reporting a member of an HOA for violating a rule/requirement ...*/
    no usages  Alperen Guncan
    @PostMapping("/report/{memberId}/{reqId}")
    public ResponseEntity<Boolean> reportUser(@PathVariable String memberId, @PathVariable long reqId,
                                              @RequestHeader(HttpHeaders.AUTHORIZATION) String token) {
        List<MembershipResponseModel> memberships =
                MembershipUtils.getActiveMembershipsForUser(memberId, token);
        if (memberships.stream().noneMatch(m -> {
            try {
                //PMD, this is silly :)
                return m.getHoaId() == requirementService.getHoaRequirement(reqId).getHoaId();
            } catch (RequirementDoesNotExist e) {
                throw new RuntimeException(e);
            }
        }))
            throw new ResponseStatusException(HttpStatus.UNAUTHORIZED, "Access is not allowed");
        hoaService.report(memberId, reqId);
        return ResponseEntity.ok( body: true);
    }

    /** Endpoint for notifying users of accepted proposals (rule changes) ...*/
    no usages  Alperen Guncan +1
    @GetMapping("/getNotifications/{memberId}/{hoaId}")
    public ResponseEntity<List<String>> getNotifications(@PathVariable String memberId,
                                                         @PathVariable long hoaId,
                                                         @RequestHeader(HttpHeaders.AUTHORIZATION) String token) {
        try {
            List<MembershipResponseModel> memberships =
                    MembershipUtils.getActiveMembershipsForUser(memberId, token);
            if (memberships.stream().noneMatch(m -> m.getHoaId() == hoaId))
                throw new ResponseStatusException(HttpStatus.UNAUTHORIZED, "Access is not allowed");
            return ResponseEntity.ok(hoaService.clearNotifications(hoaId, memberId));
        } catch (ResponseStatusException e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, e.getMessage(), e);
        }
    }
```

The code base after refactoring:

HoaController

```java
@RestController
@RequestMapping("/hoa")
public class HoaController {

    6 usages
    private transient HoaService hoaService;


    /**
     * Constructor for the HoaController.
     *
     * @param hoaService the hoa service
     */
    no usages  Rares Toader +1
    @Autowired
    public HoaController(HoaService hoaService) { this.hoaService = hoaService; }

    /**
     * Endpoint for creating an association.
     *
     * @param request             the possibly new Hoa
     * @param scheduledAfterXDays Optional int for automatic board election creation that is scheduled for X days
     * @return 200 OK if the registration is successful
     */
    no usages  Alperen Guncan +1
    @PostMapping(value = {"/create", "/create/{scheduledAfterXDays}"})
    public ResponseEntity<Hoa> register(@RequestBody HoaRequestModel request,
                                        @PathVariable Optional<Integer> scheduledAfterXDays) {
        try {
            Hoa newHoa = this.hoaService.registerHoa(request);
            if (scheduledAfterXDays.isEmpty() || scheduledAfterXDays.get() < 0) return ResponseEntity.ok(newHoa);
            Integer[] nums = Arrays.stream(LocalDateTime.now().plusDays(scheduledAfterXDays.get()) LocalDateTime
                    .format(DateTimeFormatter.ISO_DATE_TIME) String
                    .split( regex: "\\D+")).map(Integer::parseInt).toArray(Integer[]::new);
            // start automatic annual board election
            Object e = ElectionUtils.createBoardElection(new BoardElectionRequestModel(newHoa.getId(),
                    amountOfWinners: 2, List.of(),  name: "Annual board election",
                    description: "This is the auto-generated annual board election",
                    TimeModel.createModelFromArr(nums)));
            if (e != null) return ResponseEntity.ok(newHoa);
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Could not create a board election");
        } catch (Exception e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, e.getMessage(), e);
        }
    }


    /**
     * Endpoint for getting all HOAs in the database.
     *
     * @return all HOAs in the database
     */
    Rares Toader +2
    @GetMapping("/getAll")
    public ResponseEntity<List<Hoa>> getAll() {
        try {
            return ResponseEntity.ok(hoaService.getAllHoa());
        } catch (Exception e) {
            return ResponseEntity.badRequest().build();
        }
    }

    /**
     * Endpoint for getting one HOA from the database.
     *
     * @param id the id of the HOA to be retrieved
     * @return the HOA
     */
    no usages  Rares Toader +2
    @GetMapping("/getById/{id}")
    public ResponseEntity<Hoa> getById(@PathVariable long id) {
        try {
            return ResponseEntity.ok(hoaService.getHoaById(id));
        } catch (Exception e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Could not fetch HOA", e);
        }
    }


    /**
     * Setter method used when HoaService needs to be mocked
     *
     * @param h - HoaService to be mocked
     */
    1 usage  Michal Krkoska
    public void setHoaService(HoaService h) { this.hoaService = h; }


    /**
     * Endpoint for notifying users of accepted proposals (rule changes)
     *
     * @param memberId id of member to notify
     * @param hoaId    id of hoa that sends the notification
     * @param token    Authorization token used for validation
     * @return List of unread/new notifications pertaining to the member in the given hoa, if new exist
     */
    no usages  Alperen Guncan +1
    @GetMapping("/getNotifications/{memberId}/{hoaId}")
    public ResponseEntity<List<String>> getNotifications(@PathVariable String memberId,
                                                         @PathVariable long hoaId,
                                                         @RequestHeader(HttpHeaders.AUTHORIZATION) String token) {
        try {
            List<MembershipResponseModel> memberships =
                    MembershipUtils.getActiveMembershipsForUser(memberId, token);
            if (memberships.stream().noneMatch(m -> m.getHoaId() == hoaId))
                throw new ResponseStatusException(HttpStatus.UNAUTHORIZED, "Access is not allowed");
            return ResponseEntity.ok(hoaService.clearNotifications(hoaId, memberId));
        } catch (ResponseStatusException e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, e.getMessage(), e);
        }
    }
}
```

RequirementController

```java
public class RequirementController {
    6 usages
    private transient RequirementService requirementService;
    2 usages
    private final transient HoaRepo hoaRepo;

    2 usages
    private final transient HoaService hoaService;


    /** Constructor for the requirement controller ...*/
    no usages  Rares Toader
    public RequirementController(RequirementService requirementService, HoaRepo hoaRepo, HoaService hoaService) {
        this.requirementService = requirementService;
        this.hoaRepo = hoaRepo;
        this.hoaService = hoaService;
    }


    /** Gets all requirements of an HOA ...*/
    no usages  Rares Toader
    @GetMapping("/getRequirements/{hoaId}")
    public ResponseEntity<List<Requirement>> getRequirements(@PathVariable long hoaId) {
        try {
            if (hoaRepo.findById(hoaId).isEmpty())
                throw new HoaDoesntExistException("Hoa with provided id does not exist");
            return ResponseEntity.ok(requirementService.getHoaRequirements(hoaId));
        } catch (Exception e) {
            return ResponseEntity.badRequest().build();
        }
    }

    /** Adds a requirement to an HOA ...*/
    no usages  Rares Toader
    @PostMapping("/addRequirement/{hoaId}")
    @PostMapping("/addRequirement/{hoaId}")
    public ResponseEntity<Requirement> addRequirement(@PathVariable long hoaId,
                                                      @RequestBody Object prompt) {
        try {
            Requirement req = requirementService.addHoaRequirement(hoaId, String.valueOf(prompt));
            return ResponseEntity.ok(req);
        } catch (RequirementAlreadyPresent | HoaDoesntExistException e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, e.getMessage(), e);
        }
    }

    /** Removes a requirement from an HOA ...*/
    no usages  Rares Toader
    @PostMapping("/removeRequirement/{reqId}")
    public ResponseEntity<Requirement> removeRequirements(@PathVariable long reqId) {
        try {
            Requirement req = requirementService.removeHoaRequirement(reqId);
            return ResponseEntity.ok(req);
        } catch (RequirementDoesNotExist e) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, e.getMessage(), e);
        }
    }

    /** Setter method used when RequirementService needs to be mocked ...*/
    no usages  Rares Toader
    public void setRequirementService(RequirementService h) { this.requirementService = h; }

    /** Endpoint for reporting a member of an HOA for violating a rule/requirement ...*/
    no usages  Rares Toader
    @PostMapping("/report/{memberId}/{reqId}")
    public ResponseEntity<Boolean> reportUser(@PathVariable String memberId, @PathVariable long reqId,
                                              @RequestHeader(HttpHeaders.AUTHORIZATION) String token) {
        List<MembershipResponseModel> memberships =
                MembershipUtils.getActiveMembershipsForUser(memberId, token);
        if (memberships.stream().noneMatch(m -> {
```

```java
/** Endpoint for reporting a member of an HOA for violating a rule/requirement ...*/
no usages  Rares Toader
@PostMapping("/report/{memberId}/{reqId}")
public ResponseEntity<Boolean> reportUser(@PathVariable String memberId, @PathVariable long reqId,
                                          @RequestHeader(HttpHeaders.AUTHORIZATION) String token) {
    List<MembershipResponseModel> memberships =
            MembershipUtils.getActiveMembershipsForUser(memberId, token);
    if (memberships.stream().noneMatch(m -> {
        try {
            //PMD, this is silly :)
            return m.getHoaId() == requirementService.getHoaRequirement(reqId).getHoaId();
        } catch (RequirementDoesNotExist e) {
            throw new RuntimeException(e);
        }
    }))
        throw new ResponseStatusException(HttpStatus.UNAUTHORIZED, "Access is not allowed");
    hoaService.report(memberId, reqId);
    return ResponseEntity.ok( body: true);
}
}
```

8.Refactoring MemberController

Previously the MemberController contained all the endpoints that the AuthMember microservice would need. While this might sound like a good/simple idea at first, one has to keep in mind that the microservice is responsible for three main functionalities that are not directly related to one another: Manage members/password pairs, manage memberships that map members to HOAs, and manage the creation of these memberships through joining and leaving hoas. This much responsibility left to a single class made it very difficult to maintain. Some metrics that suffered severely are the WPC, which was almost 5 times the recommended upper limit, the CBO, AFTD, and NOM. Full overview of the metrics can be seen below (*LOC - 319):*

Class: MemberController

Metric	Metrics Set	Description	Value	Regular Ra...
WMC	Chidamber-...	Weighted Methods Per Class	53	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
CBO	Chidamber-...	Coupling Between Objects	18	[0..14)
RFC	Chidamber-...	Response For A Class	63	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	10	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	34	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	20	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	43	
NOM	Li-Henry M...	Number Of Methods	21	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	92	
DAC	Li-Henry M...	Data Abstraction Coupling	8	
ATFD	Lanza-Mari...	Access To Foreign Data	8	[0..6)
NOPA	Lanza-Mari...	Number Of Public Attributes	0	[0..3)
NOAC	Lanza-Mari...	Number Of Accessor Methods	4	[0..4)
WOC	Lanza-Mari...	Weight Of A Class	0.8	[0.5..1.0]
TCC	Bieman-Kan...	Tight Class Cohesion	0.6211	[0.33..1.0]
NCSS	Chr. Cleme...	Non-Commenting Source Statements	113	[0..1000)

We decided that it would be for the better if we were to split the controller into its three main aforementioned responsibilities, so the end-result is three controllers named *HOAController, MemberController*, and *MembershipController.* The improvements on all metrics across the board were drastic, as can be seen below *(LOC - 377 over 3 classes):*

Class: HOAController

Metric	Metrics Set	Description	Value	Regular Ra...
WMC	Chidamber-...	Weighted Methods Per Class	13	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
CBO	Chidamber-...	Coupling Between Objects	7	[0..14)
RFC	Chidamber-...	Response For A Class	19	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	4	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	20	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	6	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	23	
NOM	Li-Henry M...	Number Of Methods	7	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	15	
DAC	Li-Henry M...	Data Abstraction Coupling	4	
ATFD	Lanza-Mari...	Access To Foreign Data	5	[0..6)
NOPA	Lanza-Mari...	Number Of Public Attributes	0	[0..3)
NOAC	Lanza-Mari...	Number Of Accessor Methods	3	[0..4)
WOC	Lanza-Mari...	Weight Of A Class	0.5	[0.5..1.0]
TCC	Bieman-Kan...	Tight Class Cohesion	0.6	[0.33..1.0]
NCSS	Chr. Cleme...	Non-Commenting Source Statements	23	[0..1000)

Class: MemberController

Metric	Metrics Set	Description	Value	Regular Ra...
WMC	Chidamber-...	Weighted Methods Per Class	16	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
CBO	Chidamber-...	Coupling Between Objects	10	[0..14)
RFC	Chidamber-...	Response For A Class	23	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	6	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	20	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	6	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	25	
NOM	Li-Henry M...	Number Of Methods	7	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	17	
DAC	Li-Henry M...	Data Abstraction Coupling	6	
ATFD	Lanza-Mari...	Access To Foreign Data	3	[0..6)
NOPA	Lanza-Mari...	Number Of Public Attributes	0	[0..3)
NOAC	Lanza-Mari...	Number Of Accessor Methods	2	[0..4)
WOC	Lanza-Mari...	Weight Of A Class	0.6667	[0.5..1.0]
TCC	Bieman-Kan...	Tight Class Cohesion	0.6667	[0.33..1.0]
NCSS	Chr. Cleme...	Non-Commenting Source Statements	32	[0..1000)

Class: MembershipController

Metric	Metrics Set	Description	Value	Regular Ra..
WMC	Chidamber-...	Weighted Methods Per Class	34	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
CBO	Chidamber-...	Coupling Between Objects	9	[0..14)
RFC	Chidamber-...	Response For A Class	43	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	7	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	28	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	14	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	34	
NOM	Li-Henry M...	Number Of Methods	15	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	64	
DAC	Li-Henry M...	Data Abstraction Coupling	5	
ATFD	Lanza-Mari...	Access To Foreign Data	4	[0..6)
NOPA	Lanza-Mari...	Number Of Public Attributes	0	[0..3)
NOAC	Lanza-Mari...	Number Of Accessor Methods	4	[0..4)
WOC	Lanza-Mari...	Weight Of A Class	0.7143	[0.5..1.0]
TCC	Bieman-Kan...	Tight Class Cohesion	0.5824	[0.33..1.0]
NCSS	Chr. Cleme...	Non-Commenting Source Statements	72	[0..1000)

NOTE: The controller is too long to be screenshoted in a reasonable amount of space for the doc. Therefore, I suggest you look at the git commit that implements these changes instead, which can be found by the following SHA1: ***b74c33e29b8a656b3e9deb723f921b073b48ca6b***

9. MembershipService class refactor

Lastly, we decided to split the class MembershipService into two as it contained too much logic. This was identified as the "Response for Class" and "Weighted Methods Per Class". We identified that MembershipService class utilises a lot of validation logic and many methods are related solely to this purpose. Therefore, we create a new separate class MembershipValidator which contains the validation logic previously found in the MembershipService.

Original MembershipService metric:

Data Abstraction Coupling: 2
Depth Of Inheritance Tree: 1
Halstead Difficulty: 173.3906
Halstead Effort: 435180.2742
Halstead Errors: 1.9142
Halstead Length: 368
Halstead Vocabulary: 113
Halstead Volume: 2509.8259
Lack Of Cohesion Of Methods: 1
Maintainability Index: 25.4428
Message Passing Coupling: 95
Non-Commenting Source Statements: 98
Number Of Added Methods: 13
Number Of Attributes: 2
Number Of Attributes And Methods: 28
Number Of Children: 0
Number Of Methods: 14
Number Of Operations: 27
Number Of Overridden Methods: 0
Response For A Class: 69
Weighted Methods Per Class: 45
Class: MembershipService

Newly created class MembershipValidator:

```java
public final class MembershipValidator {

    /**
     * Validates the input for JoinHoaModel
     *
     * @param model JoinHoaModel model to validate
     * @return true if the name satisfies the right format
     */
    public static boolean validate(JoinHoaModel model) {
        return validateCountryCityAndStreet(model)
                && validateStreetNumber(model.getAddress().getHouseNumber())
                && validatePostalCode(model.getAddress().getPostalCode());
    }

    /**
     * Validates country, city and street of a membership
     */
    public static boolean validateCountryCityAndStreet(JoinHoaModel model) {
        return validateCountryCityStreet(model.getAddress().getCountry())
            && validateCountryCityStreet(model.getAddress().getCity())
            && validateCountryCityStreet(model.getAddress().getStreet());
    }
```

```java
/** Validates the input for country, city, and street name. It must contain ...*/
public static boolean validateCountryCityStreet(String name) {
    if (checkNullBlankEmpty(name)) {
        return false;
    }
    String trimmed = name.trim();
    return trimmed.matches( regex: "[A-Z][A-Za-z\\s]{3,49}");
}

/** Method that validates the format of a postal code. It must be of type ...*/
public static boolean validatePostalCode(String postalCode) {
    if (postalCode == null) {
        return false;
    }
    String trimmed = postalCode.trim();
    return trimmed.matches( regex: "[0-9]{4}[A-Za-z]{2}");
}

/** Validates the street number provided. Valid street numbers are the following: ...*/
public static boolean validateStreetNumber(String number) {
    if (checkNullBlankEmpty(number)) return false;
    String trimmed = number.trim();
    return trimmed.matches( regex: "^[0-9]*[A-Za-z]?$");
}

public static boolean checkNullBlankEmpty(String toCheck) {
    return (toCheck == null || toCheck.isBlank() || toCheck.isEmpty());
}
```

Refactored MembershipService metric:

Data Abstraction Coupling: 2
Depth Of Inheritance Tree: 1
Halstead Difficulty: 94.3158
Halstead Effort: 104570.7195
Halstead Errors: 0.7399
Halstead Length: 178
Halstead Vocabulary: 75
Halstead Volume: 1108.7297
Lack Of Cohesion Of Methods: 1
Maintainability Index: 32.954
Message Passing Coupling: 51
Non-Commenting Source Statements: 42
Number Of Added Methods: 10
Number Of Attributes: 2
Number Of Attributes And Methods: 25
Number Of Children: 0
Number Of Methods: 11
Number Of Operations: 24
Number Of Overridden Methods: 0
Response For A Class: 53
Weighted Methods Per Class: 16
Class: MembershipService

MembershipValidator metric:

Data Abstraction Coupling: 0
Depth Of Inheritance Tree: 1
Halstead Difficulty: 27.1818
Halstead Effort: 12030.9246
Halstead Errors: 0.175
Halstead Length: 87
Halstead Vocabulary: 34
Halstead Volume: 442.6093
Lack Of Cohesion Of Methods: 1
Maintainability Index: 41.8617
Message Passing Coupling: 26
Non-Commenting Source Statements: 17
Number Of Added Methods: 6
Number Of Attributes: 0
Number Of Attributes And Methods: 18
Number Of Children: N/A
Number Of Methods: 6
Number Of Operations: 19
Number Of Overridden Methods: 0
Response For A Class: 16
Weighted Methods Per Class: 15