Inf1B

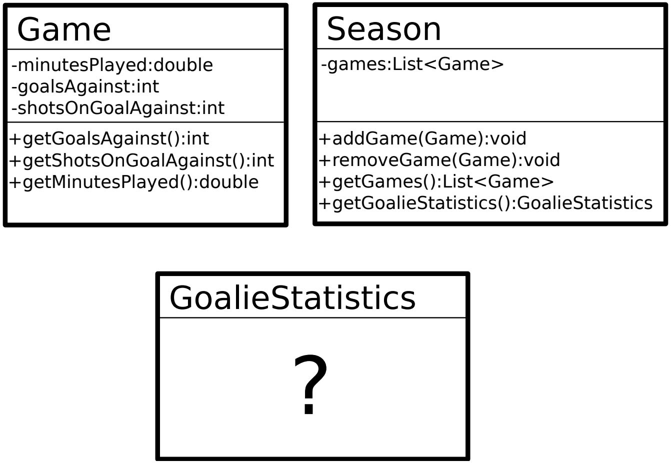
Refactoring

Fiona McNeill
adapting earlier versions by Perdita Stevens, Ewan Klein, Volker Seeker, et al.

School of Informatics

Refactoring

Hockey Analysis



Aim: Implement a class to calculate goalie statistics from all games in a season.

Hockey Analysis

- **Goals Against Average (GAA)**: Number of goals scored against a goalie divided by the number of minutes the goalie played for a season multiplied by 60 (number of minutes in a hockey game).

Hockey Analysis

- **Goals Against Average (GAA)**: Number of goals scored against a goalie divided by the number of minutes the goalie played for a season multiplied by 60 (number of minutes in a hockey game).

$$GAA = 60\left(\frac{GA}{timeplayed_{mins}}\right)$$

Hockey Analysis

- **Goals Against Average (GAA)**: Number of goals scored against a goalie divided by the number of minutes the goalie played for a season multiplied by 60 (number of minutes in a hockey game).

$$GAA = 60\left(\frac{GA}{timeplayed_{mins}}\right)$$

- **Save Percentage (SV%)**: Total number of saves (shots on goal, SOG, minus the number of goals scored on a goalie) divided by the total SOG.

Hockey Analysis

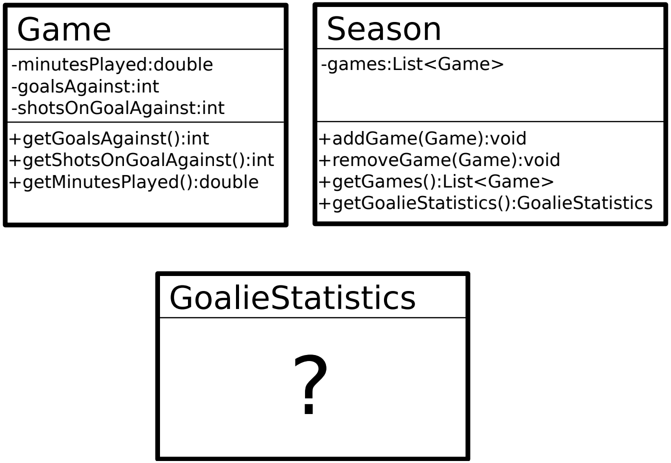
- **Goals Against Average (GAA)**: Number of goals scored against a goalie divided by the number of minutes the goalie played for a season multiplied by 60 (number of minutes in a hockey game).

$$GAA = 60\left(\frac{GA}{timeplayed_{mins}}\right)$$

- **Save Percentage (SV%)**: Total number of saves (shots on goal, SOG, minus the number of goals scored on a goalie) divided by the total SOG.

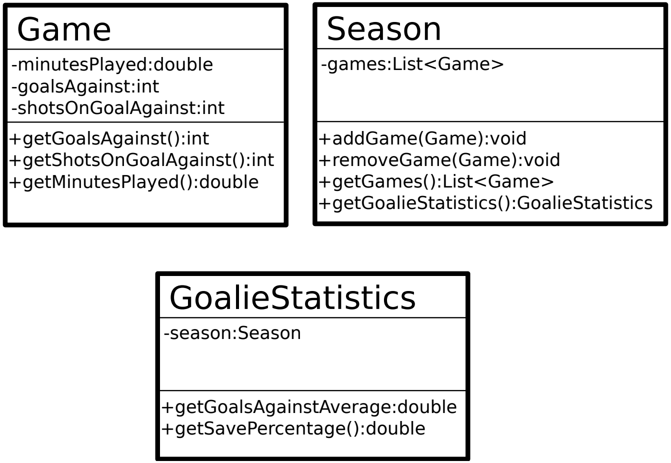
$$SV\% = \frac{SOG - Goals}{SOG}$$

Hockey Analysis



Let's make a rough plan first!

Hockey Analysis



Let's make a rough plan first!

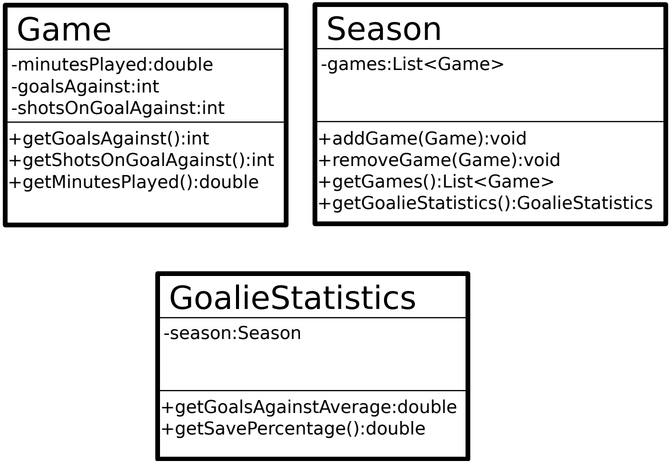
Test Driven Development

A set of unit tests is developed alongside the code, to check the expected behaviour of the code.

1. Implement test to check expected behaviour.
2. Implement feature to make test pass.

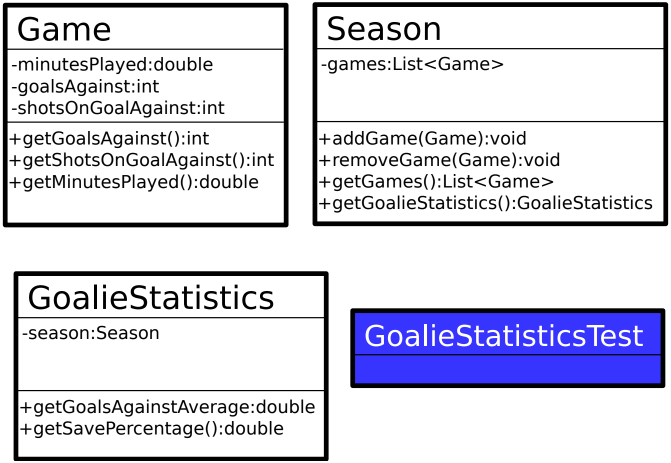
Note the order!

Hockey Analysis



We use test driven development.

Hockey Analysis



We use test driven development.

Refactoring

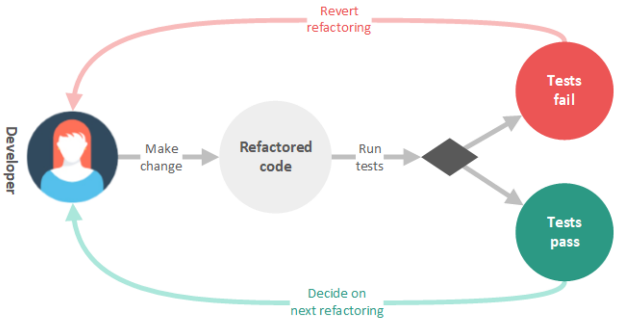
We have now successfully improved the internal structure of the code using small incremental steps whilst maintaining the original program logic.

Refactoring

We have now successfully improved the internal structure of the code using small incremental steps whilst maintaining the original program logic.

We have **refactored** the code.

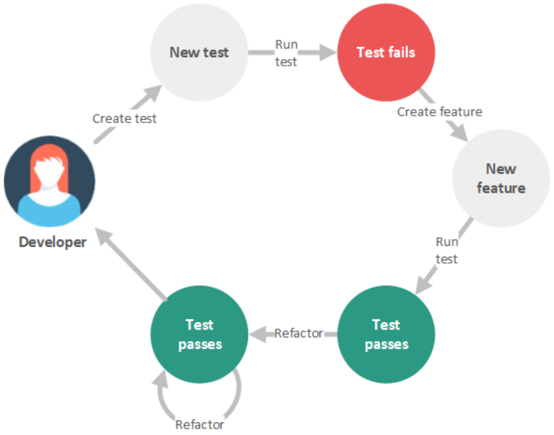
Importance of Tests



It is important that the expected behaviour of the code does not change during refactoring.

Source: `https://dzone.com/articles/what-is-refactoring`

Importance of Tests



Refactoring is part of the test driven development cycle. This does not mean the code must become perfect after every test.

Source: `https://dzone.com/articles/what-is-refactoring`

Refactoring Techniques

A large catalog of standard refactoring techniques exists, for example:

- loop splitting
- method extraction
- renaming
- ...

`https://www.refactoring.com/catalog/`

Code Smell

Identify the need for refactoring based on Warning signs in your own code.

Code Smell

A large list of common code smells exist, for example:

- Bloaters
- Object-Orientation Abusers
- Change Preventers
- . . .

`https://refactoring.guru/refactoring/smells`

Summary

- Refactoring improves the inner structure of code using small incremental steps without changing the expected behaviour
- It should be part of every development process
- With experience, you develop a nose for code smells

Reading

Books

- **How to Write Good Programs** by Perdita Stevens
- **Clean Code** by Robert Martin
- **Refactoring: Improving the Design of Existing Code** by Martin Fowler

Web Resources

- `https://www.refactoring.com/`
- `https://dzone.com/articles/what-is-refactoring`
- `https://refactoring.guru/`