MSO assignment 3, Abracha Koens (9995153) and Aron Hopman (0296163)

# **Part 1 Extending and adapting the software design**

Mermaid code in blue:

  classDiagram

    %%region Application

        namespace namespace\_Applic{

            class Application{

                +void Run**()**

                #InnerProgram AskForProgram**()**

                #void UseProgram**(**InnerProgram program**)**

                -void Execute**(**InnerProgram program**)**

                -void GetMetrics**(**InnerProgram program**)**

            }

            class ProgramImporter{

                -string path

                -string importFromtxt**(**string fileName**)**

                -bool TryFindPath**(**string fileName**,** out StreamReader output**)**

                +InnerProgram ParseProgram**(**string fileName**)**

                -Body.Builder ParseCommandBody**(**string[] code**)**

            }

            class ExamplePrograms{

                +InnerProgram basic1

                +InnerProgram basic2

                +InnerProgram advanced1

                +InnerProgram advanced2

                +InnerProgram expert1

                +InnerProgram expert2

                ...

            }

        }

        Application \*-- ProgramImporter : 1 programImporter

        ProgramImporter ..> InnerProgram : creates

        Application \*-- ExamplePrograms : 1 examplePrograms

        ExamplePrograms --> InnerProgram

        ProgramMetrics <.. Application : uses

    %%

    ProgramImporter ..> Body.Builder : creates

    ExamplePrograms ..> Body.Builder : creates

    %%region Commands

        namespace namesapce\_Commands{

            class ProgramMetrics{

                +int commandCount

                +int maxNestingLevel

                +int repeatCommandCount

            }

            class Body.Builder{

                -Body.Builder AddCommand**(**ICommand command**)**

                +Body.Builder turn**(**Dir2 dir**)**

                +Body.Builder move**(**int stepCount**)**

                +Body.Builder repeat**(**int count**,** Body.Builder body**)**

                +Body.Builder body**(**Body.Builder addedBody**)**

                +Body Build**()**

            }

            class ICommand{

                +void ApplyOnWorld**(**ref ActualWorld world**)**

                +ProgramMetrics GetMetrics**()**

            }

            class Body

            class Repeat{

                -int count

            }

            class Turn{

                -Dir2 dir

            }

            class Move{

                -int stepCount

            }

        }

        <<interface>> ICommand

        ProgramMetrics <.. ICommand : creates

        Body <-- Repeat : 1 body

        Body.Builder ..> Body : creates

        Body.Builder --> ICommand : \* commands

        Body.Builder ..> Repeat : creates

        Body.Builder ..> Turn : creates

        Body.Builder ..> Move : creates

        Body ..|> ICommand

        Body --> ICommand : \* commands

        ICommand <|.. Turn

        ICommand <|.. Move

        Repeat ..|> ICommand

    %%

    ICommand ..> ActualWorld : affects

    class InnerProgram{

        +WorldState Execute**()**

        +ProgramMetrics GetMetrics**()**

    }

    InnerProgram --> Body : 1 commands

    InnerProgram --> ActualWorld : 1 startWorld

    %%ProgramMetrics <.. InnerProgram

    %%region World

        namespace namespace\_World{

            class ActualWorld{

                +ActualWorld CopyState**()**

                +void TurnLeft**()**

                +void TurnRight**()**

                +void MoveForward**(**int dist**)**

            }

            class WorldSettings{

                %% Data that can't change while the Innerprogram is running

            }

            class WorldState{

                %% Data that can be changed by the program

                +WorldState Copy**()**

                +void TurnLeft**()**

                +void TurnRight**()**

                +void MoveForward**(**int dist**)**

                +void AddToTrace**(**IEventTrace event**)**

            }

            class PlayerState{

                +int2 pos

                +Dir4 dir

                +PlayerState Copy**()**

                +void TurnLeft**()**

                +void TurnRight**()**

                +void MoveForward**(**int dist**)**

            }

            class IEventTrace{

                +string TextualTrace**()**

            }

            class TurnTrace{

                -Dir2 dir

            }

            class MoveTrace{

                -int stepCount

            }

        }

        <<interface>> IEventTrace

        ActualWorld --> WorldState : 1 state

        ActualWorld \*-- WorldSettings

        WorldState \*-- PlayerState : 1 player

        WorldState \*-- IEventTrace : \* trace

        WorldState ..> IEventTrace : creates

        IEventTrace <|.. TurnTrace

        IEventTrace <|.. MoveTrace

    %%

    InnerProgram ..> WorldState : creates

    %%region Geometry2D

        namespace namespace\_Geometry2D{

            class Dir4{

                +Dir4 North $

                +Dir4 East $

                +Dir4 South $

                +Dir4 West $

                +int2 ToVector**()**\*

                +Dir4 Rotated**(**Dir2 dir**)**\*

                +void Rotate**(**ref Dir4 subj**,** Dir2 dir**)** $

                +int2 MovePoint**(**int2 point**,** int dist**)**

                +T Match<T>**(**T caseNorth**,** T caseEast**,** T caseSouth**,** T caseWest**)**

            }

        }

        PlayerState <-- Dir4

    %%

Design patterns:

* …

Deviations from practical 2 design:

* …

# **Part 2 Implementation and code quality**

Measures taken:

* …

Refactoring examples:

* …

Changes inspired by metrics:

* …

# **Part 3 Evaluation**

Likely future changes:

* …

High cohesion:

* …

Low coupling:

* …

# **Part 4 Testing**

Test fails + reasons:

* …

# **Part 5 Work distribution & retrospective**

Task distribution:

Part 1: …

Part 2: …

Part 3: ...

Part 4: …

What went well:

…

What could have been better:

…

What we learned:

…