COMPLETING MARS ROVER KATA

Andreas Pettersson

Alcor Academy – Running module – presentation day

03.02.2022

MARS ROVER KATA

Challenges

1. Finish the implementation
2. Implement the command pattern

```
//Challenge 1 finish the implementation
//Challenge 2 implement the Command Pattern
//    ICommand . executeOn(MarsRover or DirectionalPosition)
//        MoveCommand : ICommand
//        MoveForwardAndLeft
```

bouvet

FINISH THE IMPLEMENTATION

Where did we leave off?

```
74 references | Parajao, 1 day ago | 1 author, 4 changes
public enum Direction
{
    North,
    South,
    West,
    East
}

0 references | Parajao, 1 day ago | 1 author, 1 change
public static class DirectionExtensions
{
    private static readonly Dictionary<Direction, Direction> DirectionToTurnLeft =
        new Dictionary<Direction, Direction>
        {
            {Direction.North, Direction.West},
            {Direction.West, Direction.South},
            {Direction.South, Direction.East},
            {Direction.East, Direction.North},
        };

    private static readonly Dictionary<Direction, Direction> DirectionToTurnRight =
        new Dictionary<Direction, Direction>
        {
            {Direction.North, Direction.East},
            {Direction.East, Direction.South},
            {Direction.South, Direction.West},
            {Direction.West, Direction.North},
        };

    1 reference | Parajao, 1 day ago | 1 author, 1 change
    public static Direction TurnLeft(this Direction direction)
    {
        return DirectionToTurnLeft[direction];
    }

    1 reference | Parajao, 1 day ago | 1 author, 1 change
    public static Direction TurnRight(this Direction direction)
    {
        return DirectionToTurnRight[direction];
    }
}
```

```
[TestCaseSource(nameof(OneTurnRightScenarios))]
0 references | Andreas Pettersson, 2 hours ago | 2 authors, 4 changes
public void RoverShouldTurnRight(Tuple<DirectionalPosition, DirectionalPosition> scenario)
{
    var startPosition = scenario.Item1;
    var expectedPosition = scenario.Item2;
    var rover = new MarsRover(startPosition);
    var commands = new List<Command> { Command.TurnRight };

    rover.ExecuteCommands(commands);

    Assert.AreEqual(expectedPosition, rover.GetPosition());
}

1 reference | Parajao, 1 day ago | 1 author, 1 change
static IEnumerable<Tuple<DirectionalPosition, DirectionalPosition>> OneTurnRightScenarios()
{
    // Turn North to East
    yield return new Tuple<DirectionalPosition, DirectionalPosition>(
        new DirectionalPosition(1, 1, Direction.North), new DirectionalPosition(1, 1, Direction.East));
    // Turn East to South
    yield return new Tuple<DirectionalPosition, DirectionalPosition>(
        new DirectionalPosition(1, 1, Direction.East), new DirectionalPosition(1, 1, Direction.South));
    // Turn South to West
    yield return new Tuple<DirectionalPosition, DirectionalPosition>(
        new DirectionalPosition(1, 1, Direction.South), new DirectionalPosition(1, 1, Direction.West));
    // Turn West to North
    yield return new Tuple<DirectionalPosition, DirectionalPosition>(
        new DirectionalPosition(1, 1, Direction.West), new DirectionalPosition(1, 1, Direction.North));
}
```

```
[TestCaseSource(nameof(OneTurnLeftScenarios))]
0 references | Andreas Pettersson, 2 hours ago | 2 authors, 4 changes
public void RoverShouldTurnLeft(Tuple<DirectionalPosition, DirectionalPosition> scenario)
{
    var startPosition = scenario.Item1;
    var expectedPosition = scenario.Item2;
    var rover = new MarsRover(startPosition);
    var commands = new List<Command> { Command.TurnLeft };

    rover.ExecuteCommands(commands);

    Assert.AreEqual(expectedPosition, rover.GetPosition());
}

1 reference | Andreas Pettersson, 2 hours ago | 2 authors, 4 changes
static IEnumerable<Tuple<DirectionalPosition, DirectionalPosition>> OneTurnLeftScenarios()
{
    yield return new Tuple<DirectionalPosition, DirectionalPosition>(
        new DirectionalPosition(1, 1, Direction.North), new DirectionalPosition(1, 1, Direction.West));

    yield return new Tuple<DirectionalPosition, DirectionalPosition>(
        new DirectionalPosition(1, 1, Direction.West), new DirectionalPosition(1, 1, Direction.South));

    yield return new Tuple<DirectionalPosition, DirectionalPosition>(
        new DirectionalPosition(1, 1, Direction.South), new DirectionalPosition(1, 1, Direction.East));

    yield return new Tuple<DirectionalPosition, DirectionalPosition>(
        new DirectionalPosition(1, 1, Direction.East), new DirectionalPosition(1, 1, Direction.North));
}
```

FINISH THE IMPLEMENTATION

Executing multiple commands

```
[TestCaseSource(nameof(MultipleCommandScenarios))]
✓ | 0 references | Andreas Pettersson, 2 hours ago | 1 author, 1 change
public void ExecuteMultipleCommands(Tuple<DirectionalPosition, DirectionalPosition, List<Command>> scenario)
{
    var startPosition = scenario.Item1;
    var expectedPosition = scenario.Item2;
    var rover = new MarsRover(startPosition);
    var commands = scenario.Item3;

    rover.ExecuteCommands(commands);

    Assert.AreEqual(expectedPosition, rover.GetPosition());
}

1 reference | Andreas Pettersson, 2 hours ago | 1 author, 2 changes
static IEnumerable<Tuple<DirectionalPosition, DirectionalPosition, List<Command>>> MultipleCommandScenarios()
{
    // Move North Twice
    yield return new Tuple<DirectionalPosition, DirectionalPosition, List<Command>>(
        new DirectionalPosition(1, 1, Direction.North),
        new DirectionalPosition(1, 3, Direction.North),
        new List<Command> { Command.Move, Command.Move }
    );

    // Turn North And Move Twice
    yield return new Tuple<DirectionalPosition, DirectionalPosition, List<Command>>(
        new DirectionalPosition(1, 1, Direction.East),
        new DirectionalPosition(1, 3, Direction.North),
        new List<Command> { Command.TurnLeft, Command.Move, Command.Move }
    );

    // Turn Right Twice And Move Twice
    yield return new Tuple<DirectionalPosition, DirectionalPosition, List<Command>>(
        new DirectionalPosition(1, 3, Direction.North),
        new DirectionalPosition(1, 1, Direction.South),
        new List<Command> { Command.TurnRight, Command.TurnRight, Command.Move, Command.Move }
    );
}
```

```
6 references | Andreas Pettersson, 22 minutes ago | 2 authors, 18 changes
public class MarsRover
{
    private readonly DirectionalPosition _currentDirectionalPosition;

    5 references | ✓ 19/19 passing | Parajao, 1 day ago | 1 author, 2 changes
    public MarsRover(DirectionalPosition startDirectionalPosition)
    {
        _currentDirectionalPosition = startDirectionalPosition;
    }

    5 references | ✓ 19/19 passing | Parajao, 1 day ago | 1 author, 2 changes
    public DirectionalPosition GetPosition()
    {
        return _currentDirectionalPosition;
    }

    4 references | ✓ 15/15 passing | Andreas Pettersson, 1 hour ago | 2 authors, 11 changes
    public void ExecuteCommands(List<Command> commands)
    {
        commands.ForEach((command) =>
        {
            RunDirectionCommand(command, _currentDirectionalPosition);
        });

    }

    1 reference | Andreas Pettersson, 1 hour ago | 1 author, 1 change
    public static void RunDirectionCommand(Command command, DirectionalPosition directionalPosition)
    {
        var commandToExecute = new Dictionary<Command, Action>
        {
            { Command.TurnRight, directionalPosition.TurnRight },
            { Command.TurnLeft, directionalPosition.TurnLeft },
            { Command.Move, directionalPosition.Move },
        };

        commandToExecute[command]();
    }
```

bouvet

Implement the command pattern

First test

```
[Test]
0 references | 0 changes | 0 authors, 0 changes
public void ParseCommandAndMoveNorth()
{
    var startPosition = new DirectionalPosition(1, 1, Direction.North);
    var expectedPosition = new DirectionalPosition(1, 2, Direction.North);
    var rover = new MarsRover(startPosition);
    var commands = "M";

    rover.Execute(commands);

    Assert.AreEqual(expectedPosition, rover.GetPosition());


}
```

```
40              commandToExecute[command]();
41          }
42
            1 reference | 1/1 passing | 0 changes | 0 authors, 0 changes
43          public void Execute(string commands)
44          {
45              RunDirectionCommand(Command.Move, _currentDirectionalPosition);
46          }
47      }
48  }
```

Implement the command pattern

Scenarios and Refactoring

```
[TestCaseSource(nameof(StringCommandScenarios))]
✓ | 0 references | Andreas Pettersson, 45 minutes ago | 1 author, 1 change
public void ParseStringCommands(Tuple<DirectionalPosition, DirectionalPosition, SCommand> scenario)
{
    var startPosition = scenario.Item1;
    var expectedPosition = scenario.Item2;
    var rover = new MarsRover(startPosition);
    var commands = scenario.Item3;

    rover.Execute(commands);

    Assert.AreEqual(expectedPosition, rover.GetPosition());
}

1 reference | Andreas Pettersson, 38 minutes ago | 1 author, 2 changes
static IEnumerable<Tuple<DirectionalPosition, DirectionalPosition, SCommand>> StringCommandScenarios()
{
    yield return new Tuple<DirectionalPosition, DirectionalPosition, SCommand>(
        new DirectionalPosition(1, 1, Direction.North),
        new DirectionalPosition(1, 2, Direction.North),
        new SCommand("M"));

    yield return new Tuple<DirectionalPosition, DirectionalPosition, SCommand>(
        new DirectionalPosition(1, 1, Direction.North),
        new DirectionalPosition(1, 3, Direction.North),
        new SCommand("MM"));

    yield return new Tuple<DirectionalPosition, DirectionalPosition, SCommand>(
        new DirectionalPosition(1, 1, Direction.North),
        new DirectionalPosition(2, 1, Direction.South),
        new SCommand("MRMRM"));

    yield return new Tuple<DirectionalPosition, DirectionalPosition, SCommand>(
        new DirectionalPosition(1, 1, Direction.North),
        new DirectionalPosition(3, 3, Direction.North),
        new SCommand("MRMLMRML"));

}
```

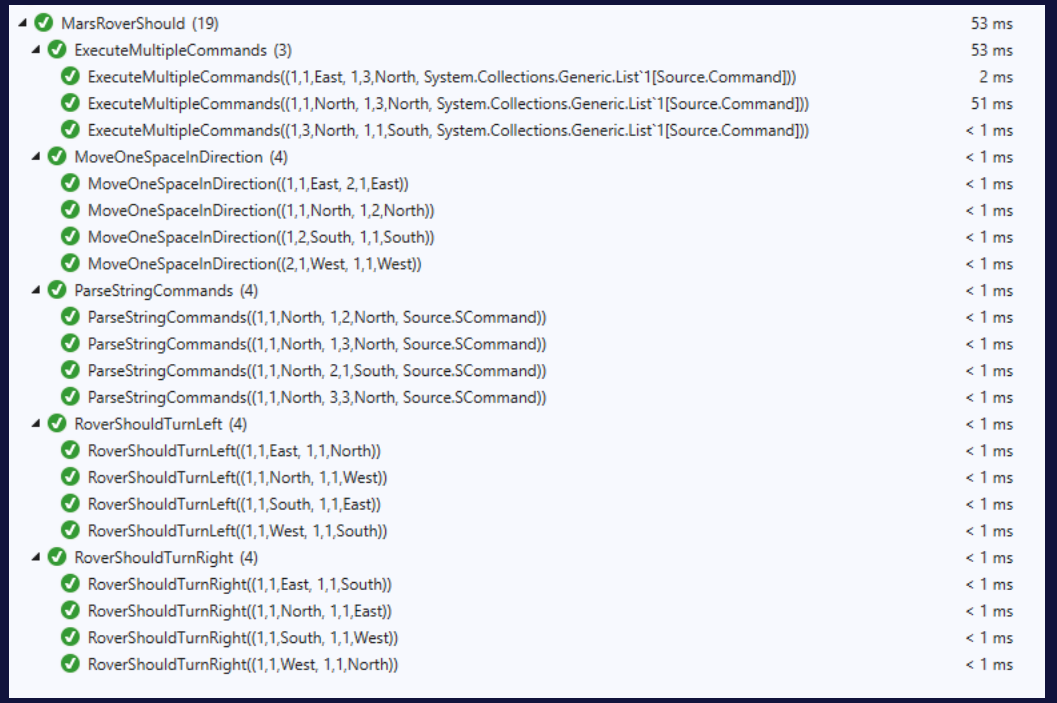
```
5 references | ✓ 19/19 passing | Parajao, 1 day ago | 1 author, 2 changes
public MarsRover(DirectionalPosition startDirectionalPosition)
{
    _currentDirectionalPosition = startDirectionalPosition;
}
```

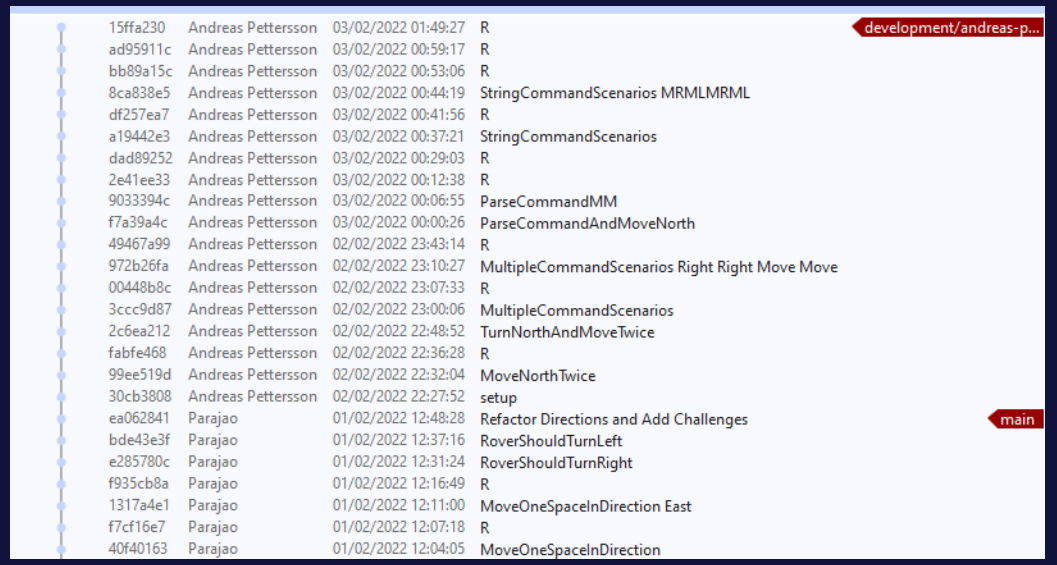
```
1 reference | ✓ 4/4 passing | Andreas Pettersson, 43 minutes ago | 1 author, 3 changes
public void Execute(SCommand commands)
{
    commands.ExecuteOn(_currentDirectionalPosition);
}
```

```
12 references | Andreas Pettersson, Less than 5 minutes ago | 1 author, 3 changes
public class SCommand
{
    private readonly string _commands;

    4 references | Andreas Pettersson, Less than 5 minutes ago | 1 author, 2 changes
    public SCommand(string commands)
    {
        this._commands = commands;
    }

    1 reference | Andreas Pettersson, 50 minutes ago | 1 author, 1 change
    public void ExecuteOn(DirectionalPosition currentDirectionalPosition)
    {
        foreach (var command in _commands)
        {
            RunCharCommand(command, currentDirectionalPosition);
        }
    }

    1 reference | Andreas Pettersson, Less than 5 minutes ago | 1 author, 2 changes
    private static void RunCharCommand(char command, DirectionalPosition directionalPosition)
    {
        var charToExecute = new Dictionary<char, Action>
        {
            { 'R', directionalPosition.TurnRight },
            { 'L', directionalPosition.TurnLeft },
            { 'M', directionalPosition.Move },
        };

        charToExecute[command]();
    }
}
```

IN CONCLUSION

- Purposeful refactoring is useful for distributing functionality.
- Code can be made more maintainable with relativly low effort.
- Mob programming improves classnames
- Version control is helpful when cutting tests.
- Questions?





bouvet

THANK YOU

andreas.pettersson@bouvet.no

GitHub.com/APettersson

bouvet