Mars Rover Kata

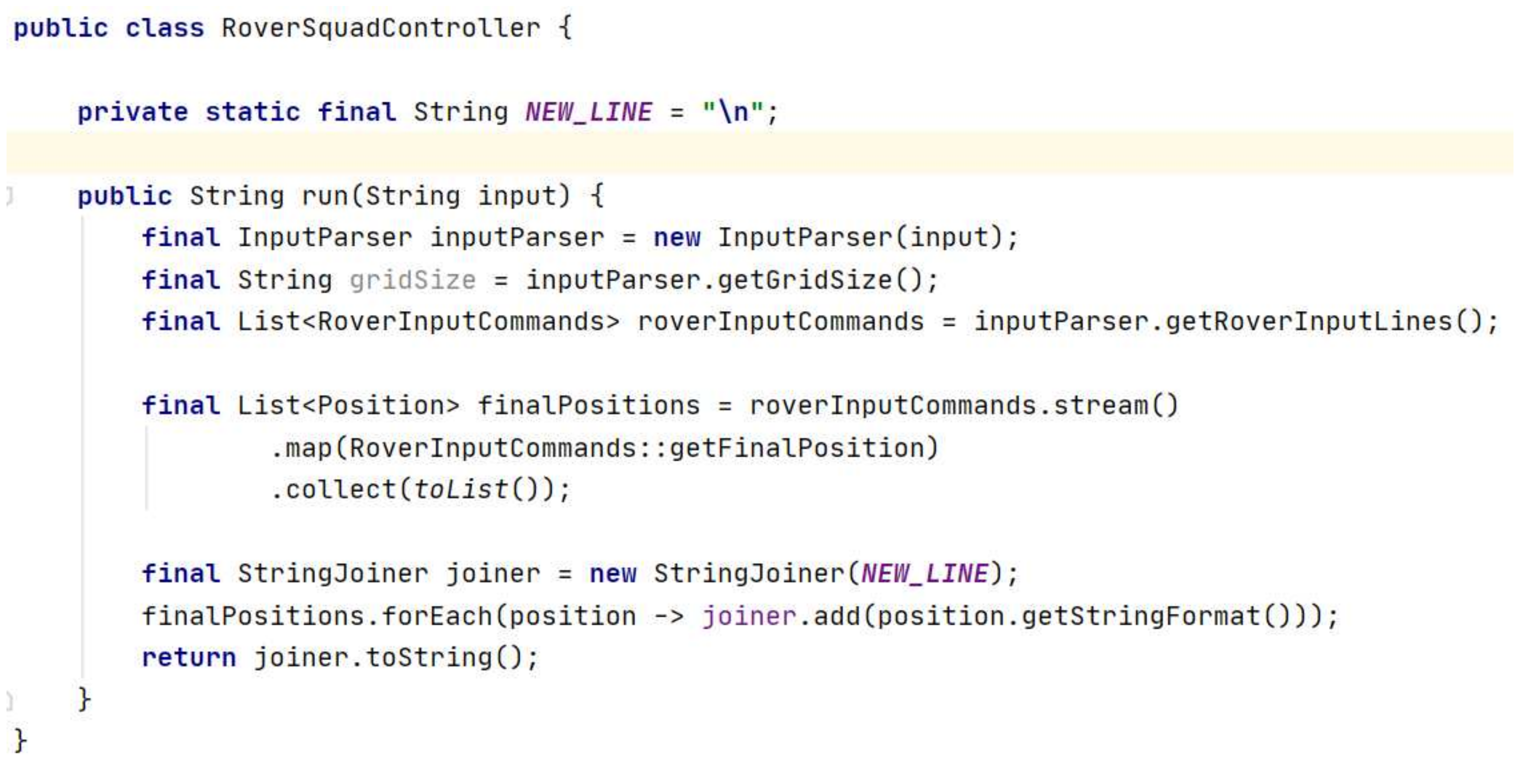
Raphael Hodel

19.11.2020

Goal

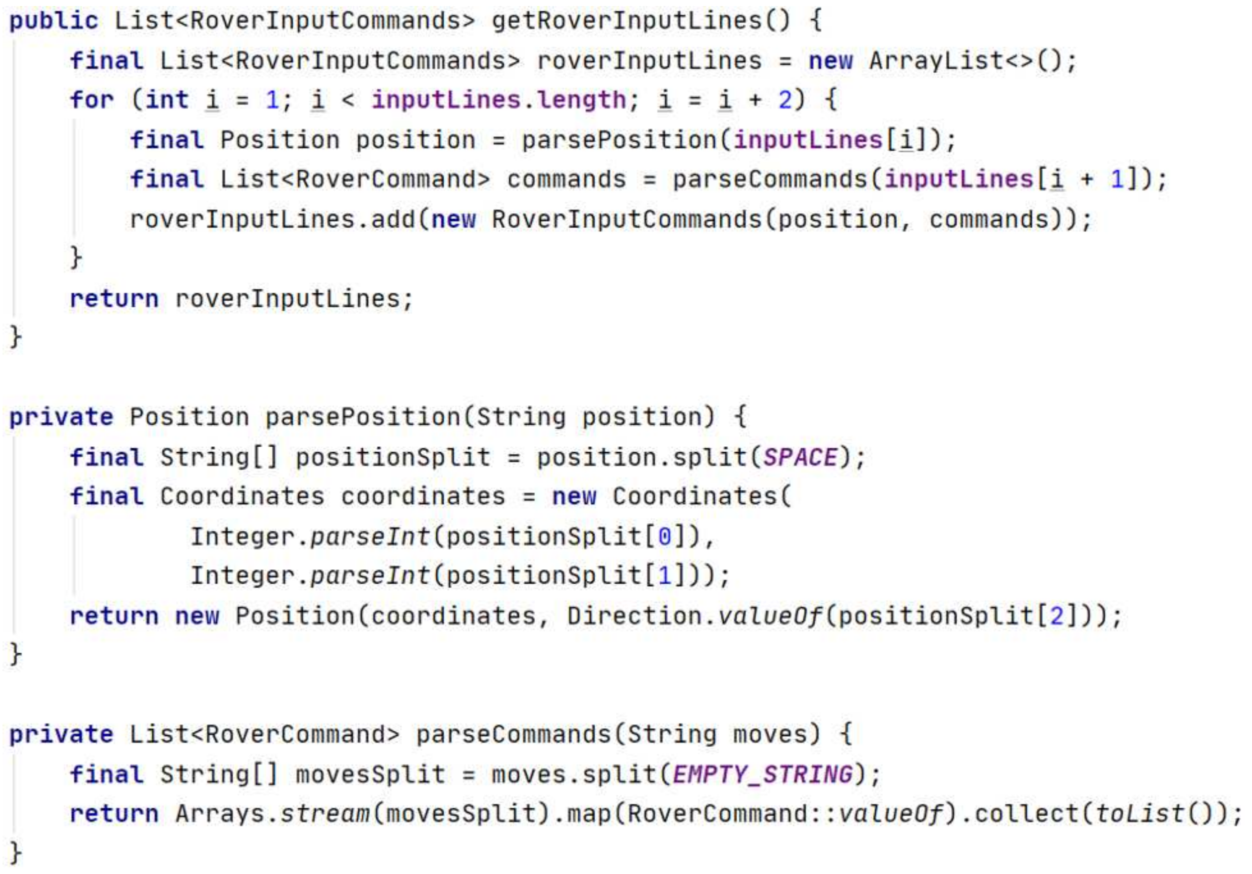
- Practicing
- Trying different things
- Having fun

RoverSquadController (main class)



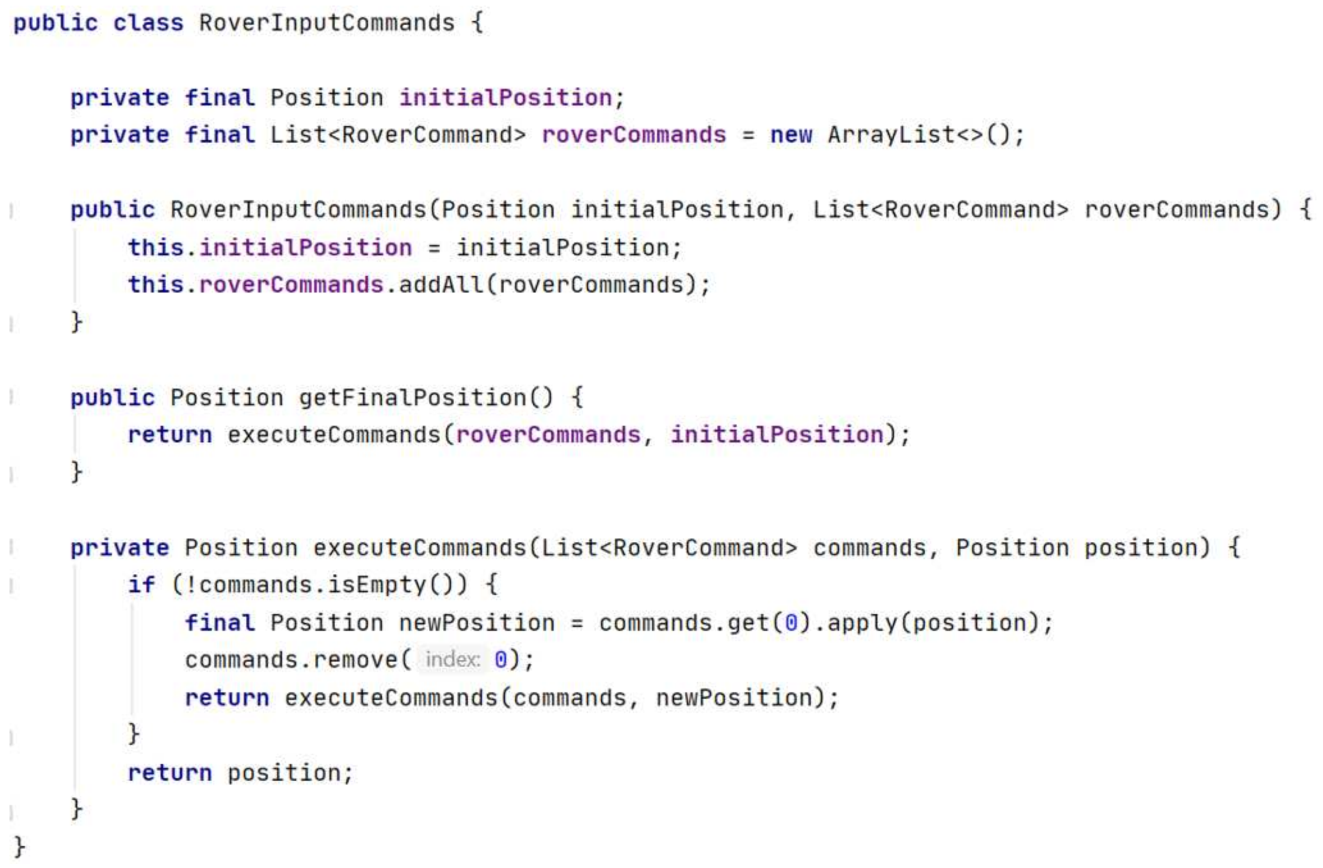
```java
public class RoverSquadController {

    private static final String NEW_LINE = "\n";

    public String run(String input) {
        final InputParser inputParser = new InputParser(input);
        final String gridSize = inputParser.getGridSize();
        final List<RoverInputCommands> roverInputCommands = inputParser.getRoverInputLines();

        final List<Position> finalPositions = roverInputCommands.stream()
                .map(RoverInputCommands::getFinalPosition)
                .collect(toList());

        final StringJoiner joiner = new StringJoiner(NEW_LINE);
        finalPositions.forEach(position -> joiner.add(position.getStringFormat()));
        return joiner.toString();
    }
}
```

InputParser



```java
public List<RoverInputCommands> getRoverInputLines() {
    final List<RoverInputCommands> roverInputLines = new ArrayList<>();
    for (int i = 1; i < inputLines.length; i = i + 2) {
        final Position position = parsePosition(inputLines[i]);
        final List<RoverCommand> commands = parseCommands(inputLines[i + 1]);
        roverInputLines.add(new RoverInputCommands(position, commands));
    }
    return roverInputLines;
}

private Position parsePosition(String position) {
    final String[] positionSplit = position.split(SPACE);
    final Coordinates coordinates = new Coordinates(
            Integer.parseInt(positionSplit[0]),
            Integer.parseInt(positionSplit[1]));
    return new Position(coordinates, Direction.valueOf(positionSplit[2]));
}

private List<RoverCommand> parseCommands(String moves) {
    final String[] movesSplit = moves.split(EMPTY_STRING);
    return Arrays.stream(movesSplit).map(RoverCommand::valueOf).collect(toList());
}
```

RoverInputCommands



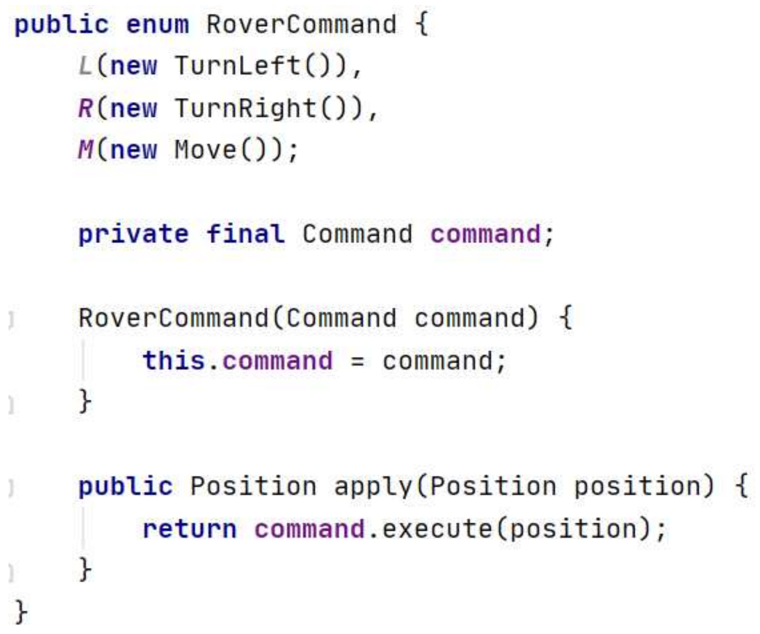
```
public class RoverInputCommands {

    private final Position initialPosition;
    private final List<RoverCommand> roverCommands = new ArrayList<>();

    public RoverInputCommands(Position initialPosition, List<RoverCommand> roverCommands) {
        this.initialPosition = initialPosition;
        this.roverCommands.addAll(roverCommands);
    }

    public Position getFinalPosition() {
        return executeCommands(roverCommands, initialPosition);
    }

    private Position executeCommands(List<RoverCommand> commands, Position position) {
        if (!commands.isEmpty()) {
            final Position newPosition = commands.get(0).apply(position);
            commands.remove( index: 0);
            return executeCommands(commands, newPosition);
        }
        return position;
    }
}
```

InitialPosition and commands belong together

Having a list of initialPositions and one of commands would be strange

ExecuteCommands is recursive → Immutability

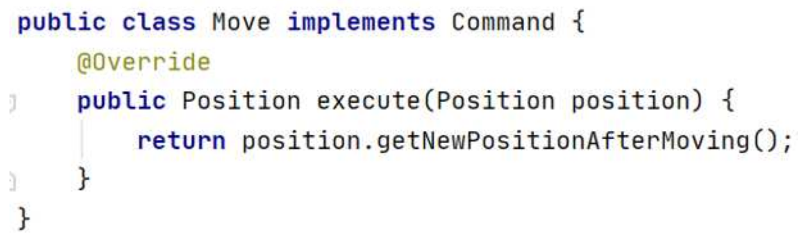
RoverCommand



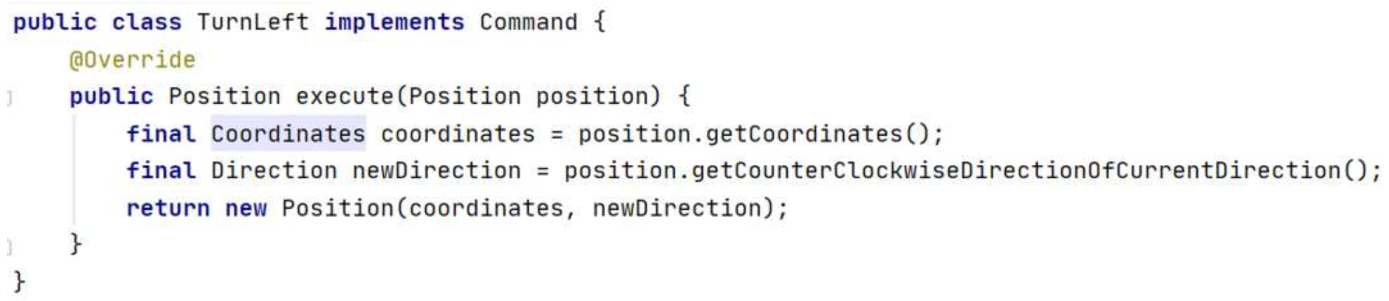
```java
public enum RoverCommand {
    L(new TurnLeft()),
    R(new TurnRight()),
    M(new Move());

    private final Command command;

    RoverCommand(Command command) {
        this.command = command;
    }

    public Position apply(Position position) {
        return command.execute(position);
    }
}
```

- Every enum entry knows what to do with the position
- Thanks Pascal :-)

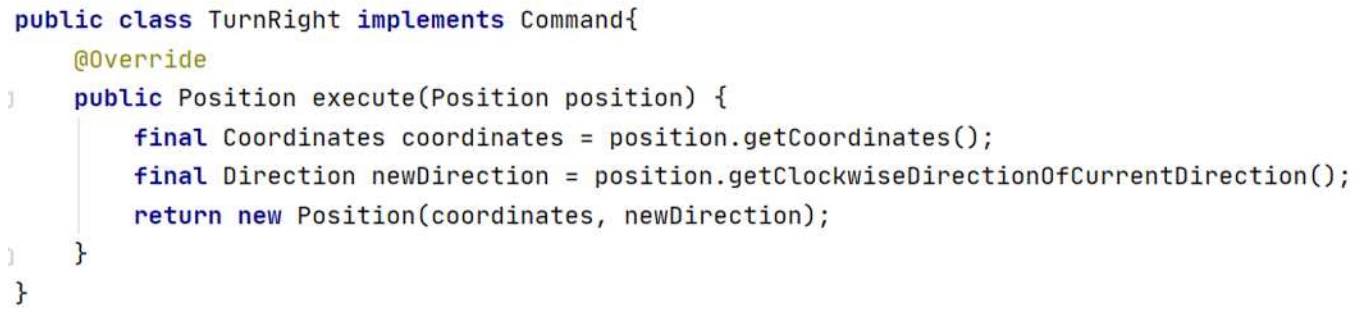
Command implementations



```java
public class Move implements Command {
    @Override
    public Position execute(Position position) {
        return position.getNewPositionAfterMoving();
    }
}
```



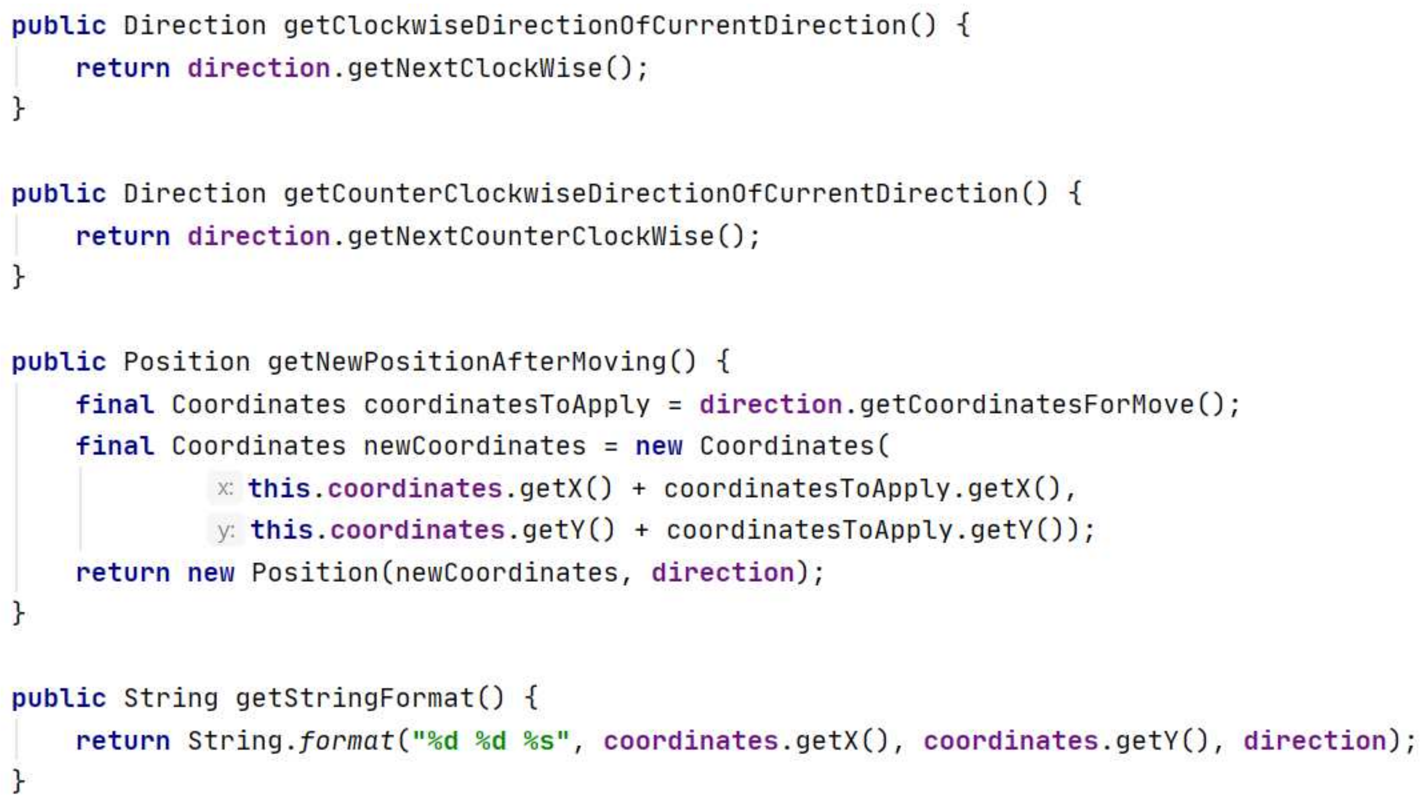
```java
public class TurnLeft implements Command {
    @Override
    public Position execute(Position position) {
        final Coordinates coordinates = position.getCoordinates();
        final Direction newDirection = position.getCounterClockwiseDirectionOfCurrentDirection();
        return new Position(coordinates, newDirection);
    }
}
```



```java
public class TurnRight implements Command{
    @Override
    public Position execute(Position position) {
        final Coordinates coordinates = position.getCoordinates();
        final Direction newDirection = position.getClockwiseDirectionOfCurrentDirection();
        return new Position(coordinates, newDirection);
    }
}
```

- Execute maybe the wrong name…
- But its the only way to stay immutable

Commands lead to Position



```java
public Direction getClockwiseDirectionOfCurrentDirection() {
    return direction.getNextClockWise();
}

public Direction getCounterClockwiseDirectionOfCurrentDirection() {
    return direction.getNextCounterClockWise();
}

public Position getNewPositionAfterMoving() {
    final Coordinates coordinatesToApply = direction.getCoordinatesForMove();
    final Coordinates newCoordinates = new Coordinates(
            x: this.coordinates.getX() + coordinatesToApply.getX(),
            y: this.coordinates.getY() + coordinatesToApply.getY());
    return new Position(newCoordinates, direction);
}

public String getStringFormat() {
    return String.format("%d %d %s", coordinates.getX(), coordinates.getY(), direction);
}
```

Conclusion

- Complete immutability → You get medals from our architect for achieving that!
- Contains recursive method → Your IQ is over 10000
- It`s all about the position. No one cares about the rover itself!
- It`s not a good but a fun implemention

Sources

https://www.businessinsider.com/favorite-curiosity-memes-2012-8?r=US&IR=T