PIPETTING ROBOT ARM SIMULATION DOCUMENTATION

1. **Synopsis**

    The purpose of this program is to simulate all the possible actions and movements of a robot arm in a square plate with fixed dimension. The movement is restricted in the square plate and the robot can only function if it is placed correctly in the beginning.

    This program is written in .NET Core 3.1 Console App using C#. There are 3 data test cases provided together with the code solution.

2. **Assumption and Justifications**
    - If the well status is full, the robot cannot drop liquid into that well. It helps to protect the well in the board from being overfull.
    - If the movement brings robot beyond the boundaries, it will stay in the same location and output error.
    - After 1 valid place command, if the second-place command is invalid, this second-place command will be ignored, and the robot can continue accepting other commands.
    - The EXIT command helps user to exit the program. This command was not mentioned in the initial program description.
    - Each well is fully filled by 1 DROP command.

3. **Note**
    - The program is started by asking user to select test data to exercise the robot. This is not mentioned in the description, but it gives user/tester the flexibility to test the program.
    - This program using Factory Design Pattern and Command Design Pattern. These design patterns help to improve the flexibility, modifiability, and maintainability of the program. For example, if developers want to add more commands into the program, it will be done easily by adding more command classes that implement the abstract class AbstractRobotCommand.
    - The ICommand interface is not required in the code, but it can be used in the future when developers want to extend the system, and they may introduce other type of command that do have common field like the AbstracRobotCommand. (See the UML diagram below for more details).

4. **UML diagram**

AbstractRobotCommand

-robotArm : RobotArm

+AbstractCommand
(RobotArm robot)
+Execute() : string

<<interface>>
ICommand

+Execute(): string

PlaceCommand

-inputParam : string

+PlaceCommand(string
inputParam, RobotArm
robotArm)
+Execute():string

DetectCommand

-plate : Plate

+DetectCommand(
RobotArm robotArm,
Plate plate)
+Execute():string

DropCommand

-plate : Plate

+DropCommand(
RobotArm robotArm,
Plat plate)
+Execute():string

MoveCommand

-inputParam : string

+MoveCommand(string
inputParam, RobotArm
robotArm)
+Execute():string

ReportCommand

-plate : Plate

+ReportCommand(
RobotArm robotArm,
Plat plate)
+Execute():string

CommandProvider

-inputParam : string
-robotArm : RobotArm
-plate : Plate

+GetCommand(): ICommand

Plate

-plate:
string[PLATE_DIMENSION,
PLATE_DIMENSION]
+PLATE_DIMENSION = 5

+IsInsideThePlate() :Bool

PipettingRobotArmProgram

-inputParam : string
-robotArm : RobotArm
-plate : Plate
-isValidStart: Bool

+Main() :void

RobotArm

-currentXLocation : int
-currentYLocation: int

+Main() :void

1..*

1..*