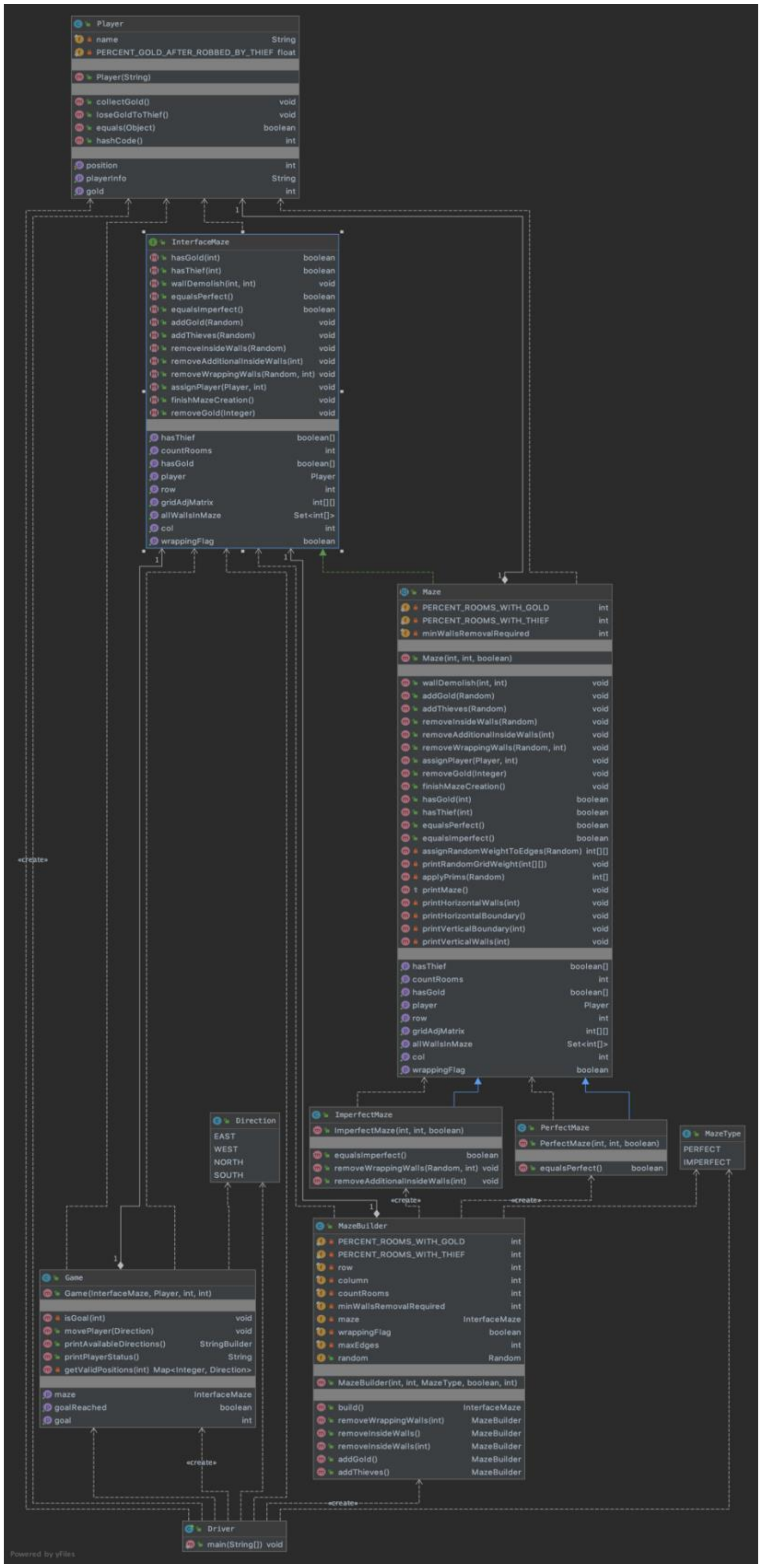
HW3 MAZE

1. UML Diagram


Player
name String
PERCENT_GOLD_AFTER_ROBBED_BY_THIEF float
Player(String)
collectGold() void
loseGoldToThief() void
equals(Object) boolean
hashCode() int
position int
playerInfo String
gold int
InterfaceMaze
hasGold(int) boolean
hasThief(int) boolean
wallDemolish(int, int) void
equalsPerfect() boolean
equalsImperfect() boolean
addGold(Random) void
addThieves(Random) void
removeInsideWalls(Random) void
removeAdditionalInsideWalls(int) void
removeWrappingWalls(Random, int) void
assignPlayer(Player, int) void
finishMazeCreation() void
removeGold(Integer) void
hasThief boolean[]
countRooms int
hasGold boolean[]
player Player
row int
gridAdjMatrix int[][]
allWallsInMaze Set<int[]>
col int
wrappingFlag boolean
Maze
PERCENT_ROOMS_WITH_GOLD int
PERCENT_ROOMS_WITH_THIEF int
minWallsRemovalRequired int
Maze(int, int, boolean)
wallDemolish(int, int) void
addGold(Random) void
addThieves(Random) void
removeInsideWalls(Random) void
removeAdditionalInsideWalls(int) void
removeWrappingWalls(Random, int) void
assignPlayer(Player, int) void
removeGold(Integer) void
finishMazeCreation() void
hasGold(int) boolean
hasThief(int) boolean
equalsPerfect() boolean
equalsImperfect() boolean
assignRandomWeightToEdges(Random) int[][]
printRandomGridWeight(int[][]) void
applyPrims(Random) int[]
printMaze() void
printHorizontalWalls(int) void
printHorizontalBoundary() void
printVerticalBoundary(int) void
printVerticalWalls(int) void
hasThief boolean[]
countRooms int
hasGold boolean[]
player Player
row int
gridAdjMatrix int[][]
allWallsInMaze Set<int[]>
col int
wrappingFlag boolean
«create»
Direction
EAST
WEST
NORTH
SOUTH
ImperfectMaze
ImperfectMaze(int, int, boolean)
equalsImperfect() boolean
removeWrappingWalls(Random, int) void
removeAdditionalInsideWalls(int) void
PerfectMaze
PerfectMaze(int, int, boolean)
equalsPerfect() boolean
MazeType
PERFECT
IMPERFECT
«create»
«create»
MazeBuilder
PERCENT_ROOMS_WITH_GOLD int
PERCENT_ROOMS_WITH_THIEF int
row int
column int
countRooms int
minWallsRemovalRequired int
maze InterfaceMaze
wrappingFlag boolean
maxEdges int
random Random
MazeBuilder(int, int, MazeType, boolean, int)
build() InterfaceMaze
removeWrappingWalls(int) MazeBuilder
removeInsideWalls() MazeBuilder
removeInsideWalls(int) MazeBuilder
addGold() MazeBuilder
addThieves() MazeBuilder
Game
Game(InterfaceMaze, Player, int, int)
isGoal(int) void
movePlayer(Direction) void
printAvailableDirections() StringBuilder
printPlayerStatus() String
getValidPositions(int) Map<Integer, Direction>
maze InterfaceMaze
goalReached boolean
goal int
«create»
«create»
Driver
main(String[]) void
1
1
1
1
1
1
Powered by yFiles

The above UML diagram represents the final class design and association between various elements like Maze, MazeBuilder, Player and Game. A maze interface will make the code more reusable and efficient. All the features that required merging of Perfecta and Imperfect maze are merged in maze abstract class. Maze Builder has been used that will take all inputs of the expected maze that should be created and will efficiently create the required maze.

We have Game as well as Person class other than maze. Person will be playing the maze. Game class encapsulates the maze and the steps of the player to solve the maze and give additional methods to move and collect gold and reach the destination in the maze.

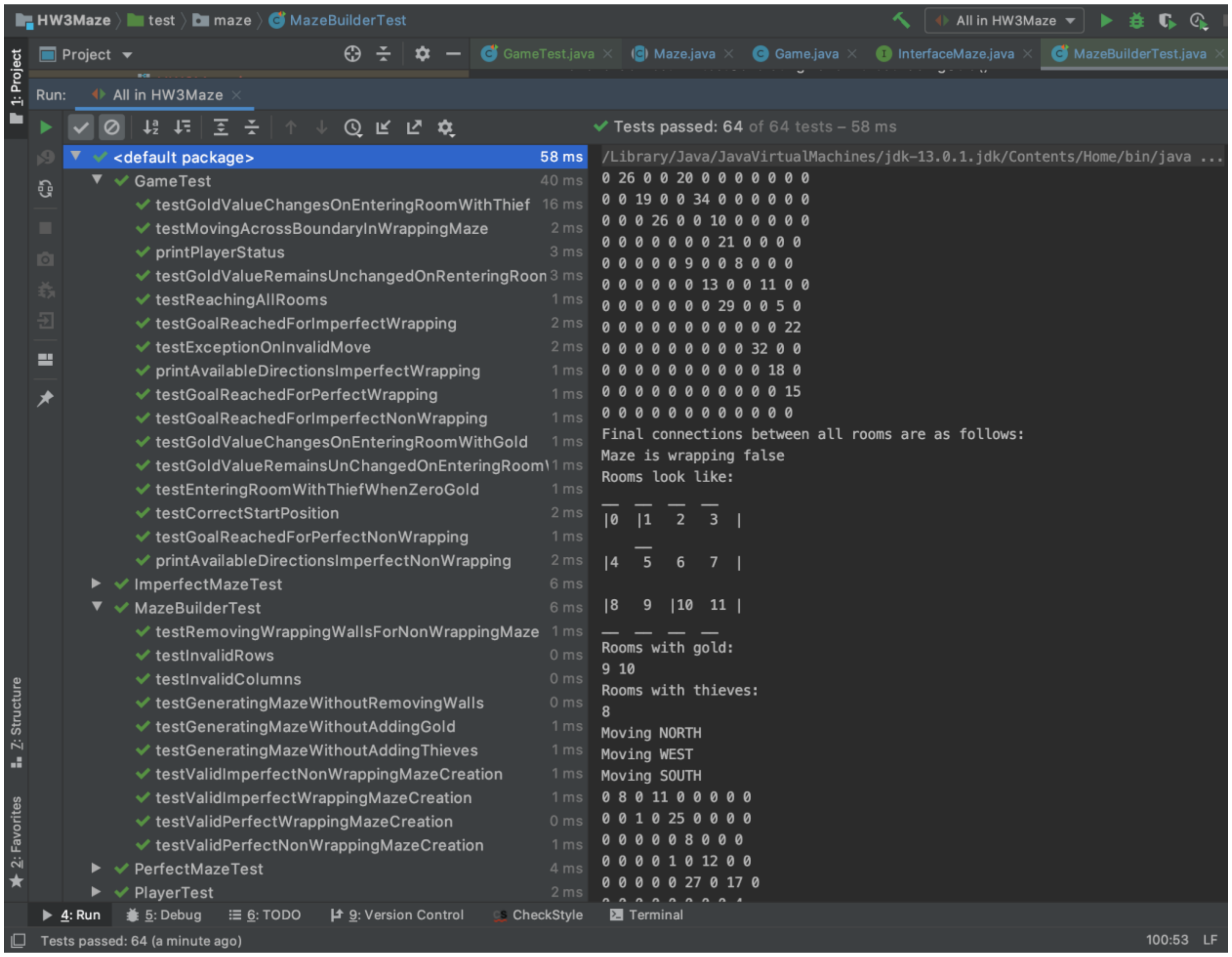
Finally, many use cases have been handled in the Driver class which allows you to execute either hardcoded sample cases for perfect, imperfect, wrapping and non-wrapping mazes or give a custom input the run the program.

2. Testing Plan

The following conditions are tested as part of the Testing

- Test for illegal argument in mazeBuilder constructor for Row, column cannot be negative.
- Test weather minimum internal walls are removed properly in mazeBuilder for both perfect and imperfect maze.
- Test weather additional internal walls are removed properly in mazeBuilder for imperfect maze.
- Test weather boundary walls are removed properly in mazeBuilder for both perfect and imperfect maze if wrapping is allowed.
- Test weather gold is assigned to appropriate rooms in maze in mazeBuilder.
- Test weather thieves are assigned to appropriate rooms in maze in mazeBuilder.
- Test weather all getter methods for Player class return proper values.
- Test weather all getter methods for Maze class return proper values
- Test weather all getter methods for Game class return proper values.
- Test for illegal argument in Game constructor for start and goal cannot be negative and should be less than total possible rooms.
- Test if correct player status is displayed at any given time.
- Test if correct directions are displayed at any given time.
- Test if player is allowed to move in correct directions in the maze.
- Test if player wins the game on reaching the goal
- Test if player is able to collect gold on entering the room with gold
- Test if player is loses gold on entering the room with thief
- Test if player's current gold value is calculated correctly.

3. Test case Execution



HW3Maze 〉 test 〉 maze 〉 MazeBuilderTest

Run: All in HW3Maze

✓ Tests passed: 64 of 64 tests – 58 ms

```
▼ ✓ <default package>                                          58 ms
  ▼ ✓ GameTest                                                 40 ms
      ✓ testGoldValueChangesOnEnteringRoomWithThief            16 ms
      ✓ testMovingAcrossBoundaryInWrappingMaze                  2 ms
      ✓ printPlayerStatus                                       3 ms
      ✓ testGoldValueRemainsUnchangedOnRenteringRoon            3 ms
      ✓ testReachingAllRooms                                    1 ms
      ✓ testGoalReachedForImperfectWrapping                     2 ms
      ✓ testExceptionOnInvalidMove                              2 ms
      ✓ printAvailableDirectionsImperfectWrapping               1 ms
      ✓ testGoalReachedForPerfectWrapping                       1 ms
      ✓ testGoalReachedForImperfectNonWrapping                  1 ms
      ✓ testGoldValueChangesOnEnteringRoomWithGold              1 ms
      ✓ testGoldValueRemainsUnChangedOnEnteringRoom\            1 ms
      ✓ testEnteringRoomWithThiefWhenZeroGold                   1 ms
      ✓ testCorrectStartPosition                                2 ms
      ✓ testGoalReachedForPerfectNonWrapping                    1 ms
      ✓ printAvailableDirectionsImperfectNonWrapping            2 ms
  ▶ ✓ ImperfectMazeTest                                         6 ms
  ▼ ✓ MazeBuilderTest                                           6 ms
      ✓ testRemovingWrappingWallsForNonWrappingMaze             1 ms
      ✓ testInvalidRows                                         0 ms
      ✓ testInvalidColumns                                      0 ms
      ✓ testGeneratingMazeWithoutRemovingWalls                  0 ms
      ✓ testGeneratingMazeWithoutAddingGold                     1 ms
      ✓ testGeneratingMazeWithoutAddingThieves                  1 ms
      ✓ testValidImperfectNonWrappingMazeCreation               1 ms
      ✓ testValidImperfectWrappingMazeCreation                  1 ms
      ✓ testValidPerfectWrappingMazeCreation                    0 ms
      ✓ testValidPerfectNonWrappingMazeCreation                 1 ms
  ▶ ✓ PerfectMazeTest                                           4 ms
  ▶ ✓ PlayerTest                                                2 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java ...
0 26 0 0 20 0 0 0 0 0 0 0
0 0 19 0 0 34 0 0 0 0 0 0
0 0 0 26 0 0 10 0 0 0 0 0
0 0 0 0 0 0 0 21 0 0 0 0
0 0 0 0 0 9 0 0 8 0 0 0
0 0 0 0 0 0 13 0 0 11 0 0
0 0 0 0 0 0 0 29 0 0 5 0
0 0 0 0 0 0 0 0 0 0 0 22
0 0 0 0 0 0 0 0 0 32 0 0
0 0 0 0 0 0 0 0 0 0 18 0
0 0 0 0 0 0 0 0 0 0 0 15
0 0 0 0 0 0 0 0 0 0 0 0
Final connections between all rooms are as follows:
Maze is wrapping false
Rooms look like:
__  __  __  __
|0  |1   2   3  |
     __
|4   5   6   7  |

|8   9  |10  11 |
__  __  __  __
Rooms with gold:
9 10
Rooms with thieves:
8
Moving NORTH
Moving WEST
Moving SOUTH
0 8 0 11 0 0 0 0 0
0 0 1 0 25 0 0 0 0
0 0 0 0 0 8 0 0 0
0 0 0 0 1 0 12 0 0
0 0 0 0 0 27 0 17 0
```

4: Run   5: Debug   6: TODO   9: Version Control   CheckStyle   Terminal

Tests passed: 64 (a minute ago)   100:53   LF