CSC120 2025S Lab No.5
Catching a Thief

Instructor: Mitsu Ogihara

1 A Game of Catching a Thief

This lab aims to write a program that plays a game of catching a thief. The game's objective is to catch a thief on a 10-by-10 grid. The user plays the role of a police officer. Initially, the thief is lat position (5,5), and the police officer is one of (1,1), (1,10), (10,1), and (10,10). The selection of the starting position is random.

In each round, the thief attempts to move to one of its eight neighbors. The thief selects one of the eight directions. If the move is impossible, the thief stays in the same position. In the same round, the user can move the office to one of its eight neighbors. The user also has the option of staying in the same position. If the selection is not possible,

The officer can apprehend the thief if, after the thief and officer move, their locational difference becomes 0 in one direction and 1 in the other direction. The user has 20 rounds to catch the thief.

2 An Example of Running the Program

Here is an example of running the program. In the visual presentation, `T` shows the location of the thief, and `P` shows the location of the police officer.

The user catches the thief in Round 7.

```
1  ------ Round = 1 ----------
2  +----------+
3  |..........|
4  |..........|
5  |..........|
6  |..........|
7  |....T.....|
8  |..........|
9  |..........|
10 |..........|
11 |..........|
12 |.........P|
13 +----------+
14 The thief is at (5,5).
15 The officer is at (10,10).
16 Enter the direction of your move.
```

```
The choices are L, R, U, D, UL, UR, DL, DR, -: UL
------ Round = 2 ----------
+----------+
|.........|
|.........|
|.........|
|....T.....|
|.........|
|.........|
|.........|
|.........|
|........P.|
|.........|
+----------+
The thief is at (4,5).
The officer is at (9,9).
Enter the direction of your move.
The choices are L, R, U, D, UL, UR, DL, DR, -: UL
------ Round = 3 ----------
+----------+
|.........|
|.........|
|.........|
|.........|
|....T.....|
|.........|
|.........|
|.......P..|
|.........|
|.........|
+----------+
The thief is at (5,5).
The officer is at (8,8).
Enter the direction of your move.
The choices are L, R, U, D, UL, UR, DL, DR, -: UL
------ Round = 4 ----------
+----------+
|.........|
|.........|
|.........|
|.........|
|.........|
|....T.....|
|......P...|
|.........|
```

```
|..........|
|..........|
+----------+
The thief is at (6,5).
The officer is at (7,7).
Enter the direction of your move.
The choices are L, R, U, D, UL, UR, DL, DR, -: -
------ Round = 5 ----------
+----------+
|..........|
|..........|
|..........|
|..........|
|..........|
|.....T....|
|......P...|
|..........|
|..........|
|..........|
+----------+
The thief is at (6,6).
The officer is at (7,7).
Enter the direction of your move.
The choices are L, R, U, D, UL, UR, DL, DR, -: L
------ Round = 6 ----------
+----------+
|..........|
|..........|
|..........|
|..........|
|..........|
|....T.....|
|.....P....|
|..........|
|..........|
|..........|
+----------+
The thief is at (6,5).
The officer is at (7,6).
Enter the direction of your move.
The choices are L, R, U, D, UL, UR, DL, DR, -: -
------ Round = 7 ----------
+----------+
|..........|
|..........|
```

```
|..........|
|..........|
|..........|
|...T......|
|.....P....|
|..........|
|..........|
|..........|
+----------+
The thief is at (6,4).
The officer is at (7,6).
Enter the direction of your move.
The choices are L, R, U, D, UL, UR, DL, DR, -: L
You've caught the thief.
--------- The final positions ---------
+----------+
|..........|
|..........|
|..........|
|..........|
|..........|
|..........|
|...TP.....|
|..........|
|..........|
|..........|
+----------+
The thief is at (7,4).
The officer is at (7,5).
```

3 The Program Structure

The program uses four static `int` variables, representing the positions of the thief and the officer. They can be named `thief_row`, `thief_column`, `police_row`, and `police_column`, or whatever names you want to use.

3.1 Presenting the locations

We use a method `present()` for this purpose.

The program can use the following static `String` constant for presenting the location as a diagram:

```
private static final String GRID_STRING =
    "+----------+\n" +
    "|..........|\n" +
```

```
"|..........|\n" +
"|..........|\n" +
"|..........|\n" +
"|..........|\n" +
"|..........|\n" +
"|..........|\n" +
"|..........|\n" +
"|..........|\n" +
"|..........|\n" +
"+----------+\n";
```

There are thirteen characters in each row.

We can visualize the positions using a `StringBuilder` object. We use a method `static void report()` for this presentation. The operation of this method.

1. Declare a `StringBuilder` variable `builder` and assign a new object instantiated with `new StringBuilder( GRID_STRING )`.
2. Obtain the position corresponding to the thief in an `int` variable, `position`. The position value is `13 * thief_row + thief_column`, and replace the single-character substring at the position with `"T"` by executing `builder.replace( position, position + 1, "T" )`.
3. Repeat the above for the officer.
4. Print the content of the builder by `System.out.print( builder )`. The print method automatically uses the `String` representing the content of the `StringBuilder` variable `builder`.
5. Report the locations of the thief and the officer.

The location variables are static. Thus, this method can access the values of the variables.

3.2 Moving the thief to a random neighbor

We use a method `move_thief()` for this purpose.

We simplify the process by separately selecting the row-wise and column-wise changes. The change amount is selected from $-1, 0, +1$. The selection can be accomplished by:

```
thief_row += (int)(Math.random() * 3) - 1;
thief_column += (int)(Math.random() * 3) - 1;
```

Then, separately for row and column, if the resulting position becomes 0, we change it back to 1, and if the resulting position becomes 11, we change it back to 10.

3.3 Moving the officer

We use a method `move_police()` for this purpose.

We prompt the user to enter the direction. The choices are: `U`, `UR`, `R`, `DR`, `D`, `DR`, `R`, `UR`, and `-`. Here, `U` is for "up," `R` is for "right," `D` is for "down," `L` is for "left," and `-` is for stationery. We store the user's response in a `String` variable `input`, and process the move in the same manner as we do for the thief. The interaction with the user occurs only in this method. We instantiate a `Scanner` object for receiving input in this method.

3.4 The main method

The main method uses a for-loop of the form:

```
for ( int round = 1; round <= 20; round ) {
...
}
```

The dotted part is filled with a code for the action to do in each round. The variable `round` represents the round number. What occurs with the use of the loop is as follows:

The value of `round` changes in sequence from 1 to 20. The dotted part is executed for each value of `round`.

The program runs with a `boolean` variable `caught`. The initial value of `caught` is `false`.

The dotted part is a big `if` statement. The condition for executing the body of this `if` statement is the value of `caught` is `false`. The body is as follows:

- Report the value of `round`.
- Call `present()`.
- Call `move_thief()`.
- Call `move_police()`.
- Update the value of `caught` with the condition whether or not the sum of the row-wise and column-wise differences between the thief and officer's positions is 1.
- If the updated value of `caught` is `true`, report that the officer has caught the thief.

After the loop terminates, call `report()` one more time to present the final positions of the thief and officer.