ปฏิบัติการนี้จะเรียนรู้เกี่ยวกับการประยุกต์ใช้ `Stack` และ `Queue`

สำหรับโจทย์แต่ละข้อ ให้มี package **`datastr`** สำหรับเก็บคลาสที่เป็น data structure และ package **`lab`** สำหรับคลาส **`Main`** ที่มี method **`main`** ที่เรียกใช้ data structure

โจทย์ปฏิบัติการ ข้อ 1

ในคลาส **`Main`** เขียน recursive method **`reverse`** ที่คืนค่าเป็นสตริงที่เรียงตัวอักขระใน explicit parameter ย้อนจากหลังไปหน้า และสร้างคลาส **`ARI`** ที่ใช้เก็บ activation record instance ของ method **`reverse`** โดยให้เก็บเฉพาะตัวแปร local กับ parameter ของ method นี้ (ไม่ต้องเก็บ return address)

นอกจากนั้นให้ใช้คลาส **`Stack`** ที่เรียนมาเพื่อสร้าง stack เพื่อจำลองการเก็บ activation record instance ของ method **`reverse`** ระหว่างการทำงาน พิจารณาว่าในการทำงานของ method **`reverse`** นี้ จะต้อง push และ pop ตัว activation record instance กับ stack เมื่อใดแล้วแทรกคำสั่งที่ให้แก้ค่าใน stack และแก้ค่าใน activation record instance ให้ถูกต้อง

แสดงผลลัพธ์เป็นการเปลี่ยนค่าใน stack ของ ARI ดังตัวอย่างนี้

ตัวอย่างการทำงาน สำหรับ `reverse("12345")`

```
Call reverse("12345")
=== BOTTOM OF STACK ===
s     = 12345
back  =
backR =
------------------------
```

```
Call reverse("2345")
=== BOTTOM OF STACK ===
s     = 12345
back  = 2345
backR =
------------------------
s     = 2345
back  =
backR =
------------------------
```

```
Call reverse("345")
=== BOTTOM OF STACK ===
s     = 12345
back  = 2345
backR =
------------------------
s     = 2345
back  = 345
backR =
------------------------
s     = 345
back  =
backR =
------------------------
```

```
Call reverse("45")
=== BOTTOM OF STACK ===
s     = 12345
back  = 2345
backR =
------------------------
s     = 2345
back  = 345
backR =
------------------------
s     = 345
back  = 45
backR =
------------------------
s     = 45
back  =
backR =
------------------------
```

```
Call reverse("5")
=== BOTTOM OF STACK ===
s     = 12345
back  = 2345
backR =
------------------------
s     = 2345
back  = 345
backR =
------------------------
s     = 345
back  = 45
backR =
------------------------
s     = 45
back  = 5
backR =
------------------------
s     = 5
back  =
backR =
------------------------
```

```
Return from reverse("5")
=== BOTTOM OF STACK ===
s     = 12345
back  = 2345
backR =
------------------------
s     = 2345
back  = 345
backR =
------------------------
s     = 345
back  = 45
backR =
------------------------
s     = 45
back  = 5
backR =
------------------------
s     = 5
back  =
backR =
------------------------
```

```
Return from reverse("45")
=== BOTTOM OF STACK ===
s     = 12345
back  = 2345
backR =
------------------------
s     = 2345
back  = 345
backR =
------------------------
s     = 345
back  = 45
backR =
------------------------
s     = 45
back  = 5
backR = 5
------------------------
```

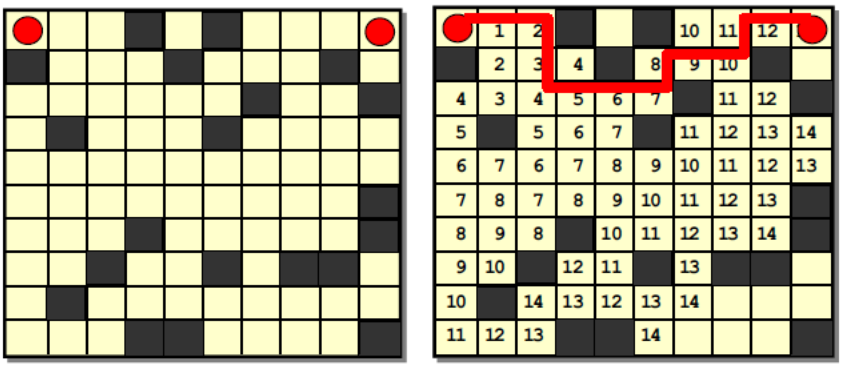
```
Return from reverse("345")
=== BOTTOM OF STACK ===
s     = 12345
back  = 2345
backR =
------------------------
s     = 2345
back  = 345
backR =
------------------------
s     = 345
back  = 45
backR = 54
------------------------
```

```
Return from reverse("2345")
=== BOTTOM OF STACK ===
s     = 12345
back  = 2345
backR =
------------------------
s     = 2345
back  = 345
backR = 543
------------------------
```

```
Return from reverse("12345")
=== BOTTOM OF STACK ===
s     = 12345
back  = 2345
backR = 5432
------------------------
```

โจทย์ปฏิบัติการ ข้อ 2

จากหนังสือ https://www.cp.eng.chula.ac.th/books/ds-vjjv/ เราสามารถใช้ queue ในการหาวิถีที่สั้นที่สุดที่จะเดินจากช่องหนึ่งในตาราง ไปยังอีกช่องดังแสดงในรูปข้างล่าง



รูปที่ 7-7 การหาวิถีสั้นสุดในตารางที่มีบางช่องเป็นสิ่งกีดขวาง

ในในคลาส **Lee** นี้ มี method **findPath** ที่ใช้ **ArrayQueue** เพื่อหาเส้นทางหรือที่เรียกว่า วิถี (path)

```
01 public class Lee {
02   private static final int SPACE = -1;
03   private static final int BLOCK = -9;
04   private static int[][] map = new int[10][10];
05
06   private static class Pos {
07     int row, col;
08     Pos(int r, int c) {row = r; col = c;}
09   }
10   public static void main(String[] args) {
11     for (int i = 0; i < map.length; i++)
12       for (int j = 0; j < map[i].length; j++)
13         map[i][j] = Math.random()<0.2 ? BLOCK : SPACE;
14     findPath(new Pos(0,0), new Pos(0,map[0].length-1));
15     for (int i = 0; i < map.length; i++) {
16       for (int j = 0; j < map[i].length; j++)
17         System.out.printf("%4d", map[i][j]);
18       System.out.println();
19     }
20   }
21   static void findPath(Pos source, Pos target) {
22     map[source.row][source.col] = 0;
23     map[target.row][target.col] = SPACE;
24     Queue q = new ArrayQueue();  q.enqueue(source);
25     while (!q.isEmpty()) {
26       Pos p = (Pos) q.dequeue();
27       if (p.row == target.row && p.col == target.col) break;
28       expand(q, p.row + 1, p.col, map[p.row][p.col]);
29       expand(q, p.row - 1, p.col, map[p.row][p.col]);
30       expand(q, p.row, p.col + 1, map[p.row][p.col]);
31       expand(q, p.row, p.col - 1, map[p.row][p.col]);
32     }
33   }
34   static void expand(Queue q, int r, int c, int k) {
35     if (r < 0 || r >= map.length ||
36         c < 0 || c >= map[r].length ||
37         map[r][c] != SPACE) return;
38     map[r][c] = k + 1;
39     q.enqueue(new Pos(r, c));
40   }
41 }
```

คลาสภายในใช้เก็บตำแหน่งของช่องในตาราง

สุ่มใส่สิ่งกีดขวาง

หาวิถีสั้นสุด

แสดงตารางทางจอภาพ

เลิกค้นเมื่อพบตำแหน่งเป้าหมาย

แผ่อาณาเขตการค้นตามแนวกว้างไปทั้งสี่ทิศ

ถ้าตกขอบหรือไม่ใช่ช่องว่างก็ไม่เติม

เพิ่มระยะสั้นสุดขึ้นอีก 1 ให้กับช่องนี้และนำตำแหน่งของช่องนี้ใส่แถวคอย

รหัสที่ 7-9 โปรแกรมหาวิถีสั้นสุดในตาราง โดยใช้การค้นตามแนวกว้าง

ให้ทำความเข้าใจโปรแกรมนี้ แล้วนำไปทดลองเพื่อหาวิถีที่สั้นที่สุด จากนั้นให้ใช้คลาส **Stack** แทนคลาส **Queue** ในการหาวิถี (path) แล้วดูว่าวิถี (path) ที่ได้มานี้แตกต่างจากเดิมอย่างไร เพราะเหตุใด