Just Another Race

In a range there are cats, mice, and balls. The cats are interested in the balls (because balls have nice colors and the cats like to play with them) and are also tempted by the mice which they would like to eat. We are in possession of a situation map of the range, with special marking for important elements. Here are the details:

- Everything happens in *steps*.
- A step is taken when the user presses the 'Enter' key. First, the mice move, and then the cats.
- The process ends when there are no more mice, or every cat has grabbed a ball and stopped, or there are no more balls to grab.

1. **Cats**, marked with capital **C** letters try get a ball to play with, and will normally go to grab the closest ball. But if they sniff a mouse which is closer, they will be attracted by the mouse, and move toward the nearest mouse they sniffed to grab it. A cat moves 1 square per step in one of the directions N, NE, E, SE, S, SW, W, NW. If a cat catches a mouse, both the cat and the mouse will disappear from the range. If it grabs a ball – possible when they reach on a ball position – the ball will disappear, and the cat will stop moving, enjoying the ball from that step onwards.
2. **Mice,** marked with capital **M** letters, will move towards the nearest ball. When they are close to al ball, (like the mouse near ball **5**, or the mouse near ball **c**, in Fig. 1) they do not move. A mouse moves 1 square per step in one of four possible directions N, E, S, W.
3. **Balls**, symbolized with hex digits (from **0** to **f**, only lowercase letters are used), can be caught by the cats, and they also attract mice. They do not move at all and disappear once grabbed by a cat.

At the beginning of the game, and after each step you should print to **System.out** – *text mode* – the current situation map – see Fig. 1 below.

There is a package provided, called **oop.range** which contains in class **Provided** the following public static methods:

- **public static Coord[] readElement(char[][] map, char symbol)** returns an array with coordinates for the specific symbol character.
- **public static Coord[] readBalls(char[][] map)** returns an array with all ball coordinates.
- **public static Coord[] getClosest(Coord me, char[][] map, char type)** returns an array of closest coordinates to **me** sorted in ascending order**.** The **type** parameter can be '**M**' for mouse or '**B**' for Ball. If no elements of that type exist, it returns **null**.
- **public static Coord calculateNextPosition(char[][] map, char myType, char targetType, Coord myPos, Coord targetPos)** returns the coordinate of the move position towards the position **pos**, for an object located at position **myPos**, executable in one step. The **myType** parameter can be '**M**' meaning for mouse or '**C**' for cat. The **targetType** parameter can be '**M**' for mouse or '**B**' for ball.
- **public static char[][] readMap(String path)** returns an array with the map read from a file whose path is given as an argument to the method.

The package **oop.range** also includes the instantiable class **Coord**, with integer fields named **row** and **column** and a double field named **distance** and setters and getters for those fields**.** Rows and columns are numbered starting at zero. **Coord** has two constructors, one without parameters, which sets all fileds to zero, and one with two parameters **Coord(int row, int column).** The **distance** field is updated by the **getClosest** and **calculateNextPosition** methods when invoked for the **Coord** objects they create and return.

You should use what is provided in the package in your implementation, not implement them yourself.

An example start configuration, shown in Fig. 1, contains the size of the range, and the situation map. The start configuration is read from a file named given as an argument to main**.**

Draw the *class diagram* and develop a *java standalone* program to simulate this game. Don't forget to briefly document it.

```
13 86
..............................................................................
...C.........................................................7................
.....................................................................8........
.............................................................6................
..........0.......1.....................5.....................................
....................2.....................M.....................................
.....................................................................M.....9...
..........C..............M.....4.....................................................
..............................................................................
...............M..........3.....................c.....................a.............
....C.....................................M.....................................
.....................................C.....................b.....................
.....................................d.....................................................
```

Fig. 1 Example of a possible start configuration. The map you should print in every step does not include first line.