	Carátula para entrega de prácticas
Facultad de Ingeniería	Ingeniería en Computación

Profesor(a):	
Asignatura:	Programación Orientada a Objetos
Grupo:	1
No de Práctica(s):	2
Integrante(s):	11700202-9
No. de Equipo de cómputo empleado:	
Semestre:	2024-1
Fecha de entrega:	10 de septiembre de 2023
Observaciones:	

CALIFICACIÓN: ______

Index.

Practice 2 - Fundamentals and language elements

32x32 Chess pieces moves.

1.- Introduction.

This report has the objective of explaining the second practice of the Object oriented programming subject, in which we must explore other features and functions of Java.

We must build an algorithm for showing the chess pieces moves on a 32x32 grid, asking the user for the initial position and based on the user choice, we have to show the possible moves without any exception or error getting out of the grid.

1.1.- Hypothesis

We can make the grid as a 32x32 character matrix, with blank spaces on the unused positions of the grid, the initial letter from the piece's name to represent the position and asterisks for possible moves.

Also we need to take care of the limits of the grid, because for the majority of pieces, we will use loops. The pieces' limitations and conditions in function of their positions. And for possible wrong inputs given by the user.

2.- Development

The steps to follow were:

- Create a menu for selecting the piece to show, asking for initial conditions of the piece, like the side or the initial position in case of the pawn.
- Validate the coordinate inputs for avoiding errors.
- Build the matrix (O(32^2) time complexity)
- Once selected an option, draw the moves of the piece.
    - For the horse/knight, we check possible L moves, validating 8 moves (O(1) Time complexity)

    - For the bishop, iterate diagonally until we get to the grid limits for all quadrants. O(n) complexity; n<=31.
    - For the tower, iterate in a straight line until we get to the grid limits for the four quadrants. O(n) complexity; n<=31.
    - For the king, we check the possible moves on a radius of one O(1) complexity.
    - For the queen, we use the same logic of the tower and the bishop O(n + m) n being diagonal moves, n <= 31, m being straight moves m<=31.
    - For the pawn, asking for the side (black or white), then we check if the piece is at an initial position, if it's true, we color two moves, if it isn't, we color a single move. O(1) complexity.
- Print the grid, divided by " | " character for marking the positions division.

Grid reference system.

(0,0) is the left top corner.

(31,31) is the bottom right corner.

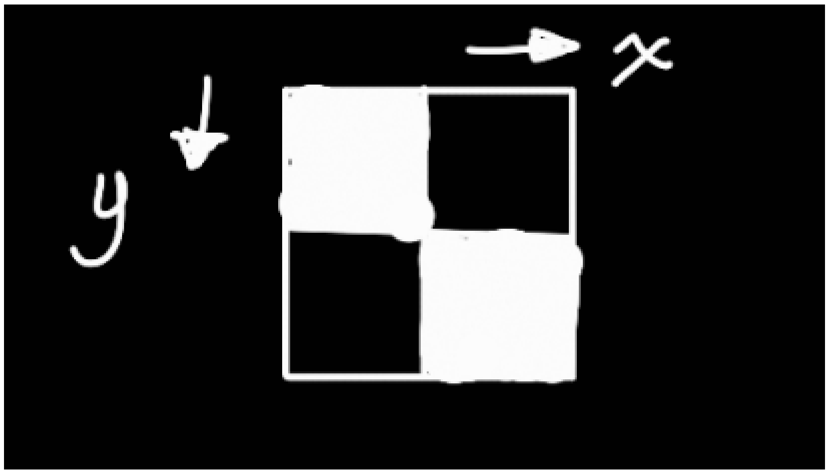


Figure 1 (Grid reference system).

Pawn logic.

- If it's at an initial position, the pawn can move two steps forward. Otherwise, the pawn could only move one step forward.

- Pawn could only capture in a diagonal way but only if the other piece is +1 position forward and in diagonal

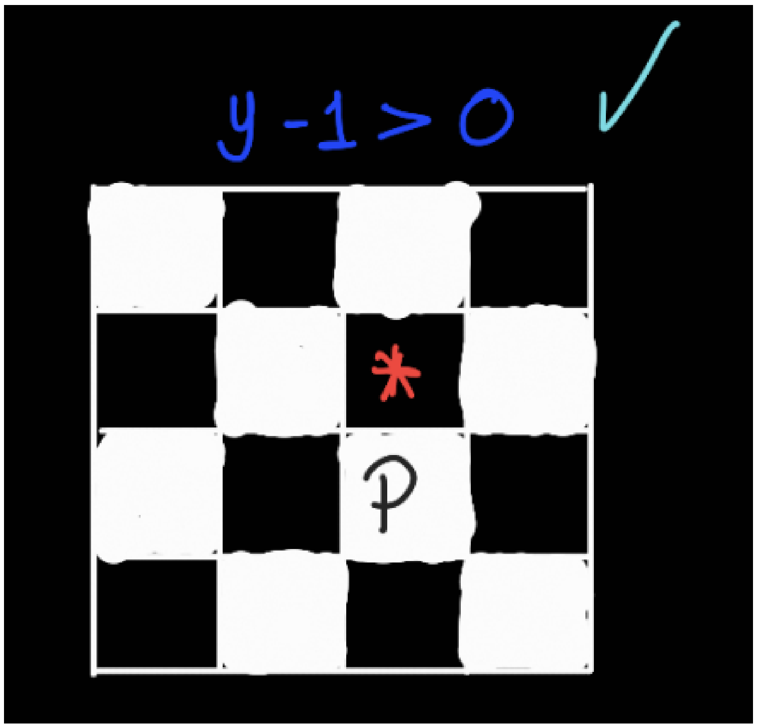
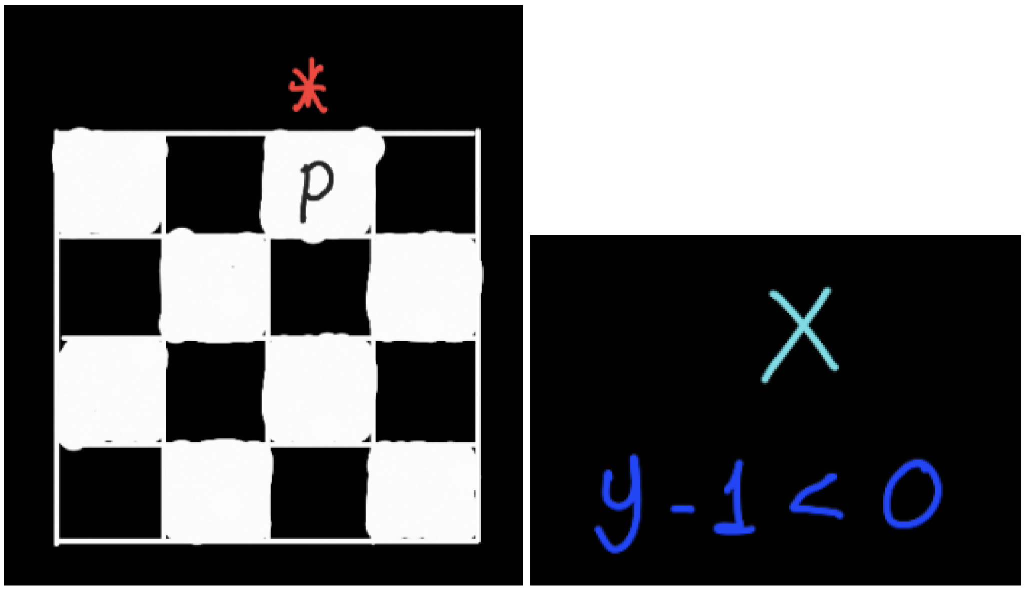
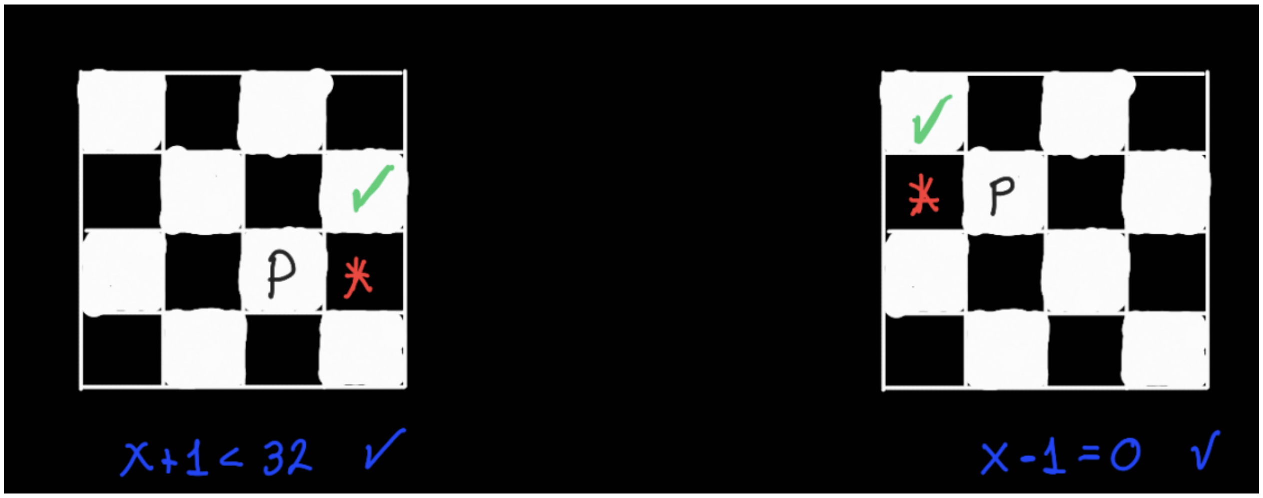


Figure 2 (Legal pawn single move).



Figures 3 and 4 (Illegal pawn move).



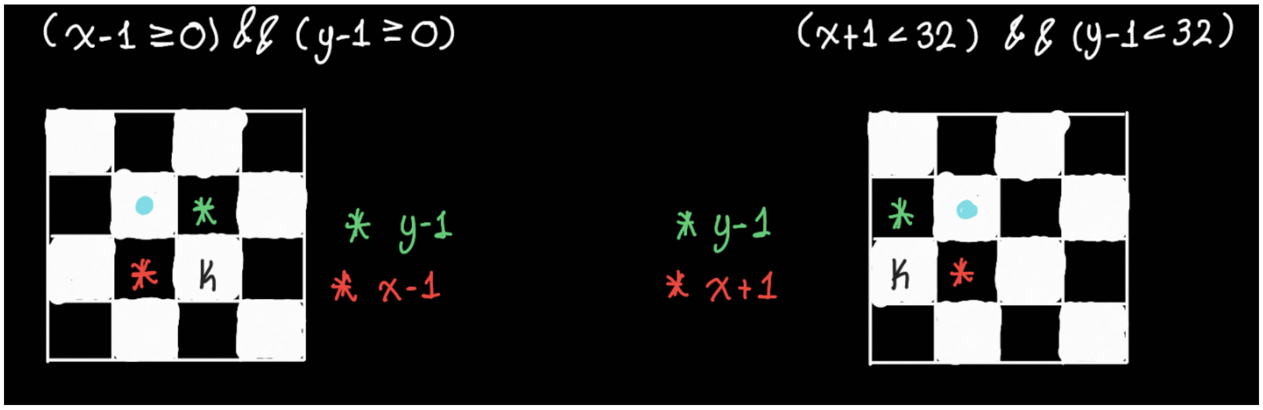
Figures 5 and 6 (Legal pawn capture moves).

Pseudocode:

```
Case 6:
    Set tablero[y][x] = 'P'
    if bando == 0 then
        if y - 1 >= 0 then
            Set tablero[y - 1][x] = 'x'
            if y == 31 then
                Set tablero[y - 2][x] = 'x'
            end if
            if x - 1 >= 0 then
                Set tablero[y - 1][x - 1] = 'x'
            end if
            if x + 1 < 32 then
                Set tablero[y - 1][x + 1] = 'x'
            end if
        end if
    else if bando == 1 then
        if y + 1 < 32 then
            Set tablero[y + 1][x] = 'x'
        end if
        if y + 1 < 32 and x - 1 >= 0 then
            Set tablero[y + 1][x - 1] = 'x'
        end if
        if y + 1 < 32 and x + 1 < 32 then
            Set tablero[y + 1][x + 1] = 'x'
        end if
        if y == 0 then
            Set tablero[y + 2][x] = 'x'
        end if
    end if
```

King logic.

King could move one position diagonally, in any way.


$(x-1 \geq 0)$ && $(y-1 \geq 0)$
$(x+1 < 32)$ && $(y-1 < 32)$
K
✱ y-1
✱ x-1
✱ y-1
✱ x+1
K

Figures 7 and 8 (King legal moves).

Pseudocode:

```
Set tablero[y][x] = 'K'
if y - 1 >= 0 then
    Set tablero[y - 1][x] = 'x'
end if
if y + 1 < 32 then
    Set tablero[y + 1][x] = 'x'
end if

if x - 1 >= 0 then
    Set tablero[y][x - 1] = 'x'
end if
if x + 1 < 32 then
    Set tablero[y][x + 1] = 'x'
end if
```

```
if y - 1 >= 0 and x + 1 < 32 then
    Set tablero[y - 1][x + 1] = 'x'
end if
if y - 1 >= 0 and x - 1 >= 0 then
    Set tablero[y - 1][x - 1] = 'x'
end if
if y + 1 < 32 and x - 1 >= 0 then
    Set tablero[y + 1][x - 1] = 'x'
end if
if y + 1 < 32 and x + 1 < 32 then
    Set tablero[y + 1][x + 1] = 'x'
end if
```

Horse/knight logic.

- Horse moves, describing an L, two steps forward and then one in a different direction or one step forward and then two in a different direction.

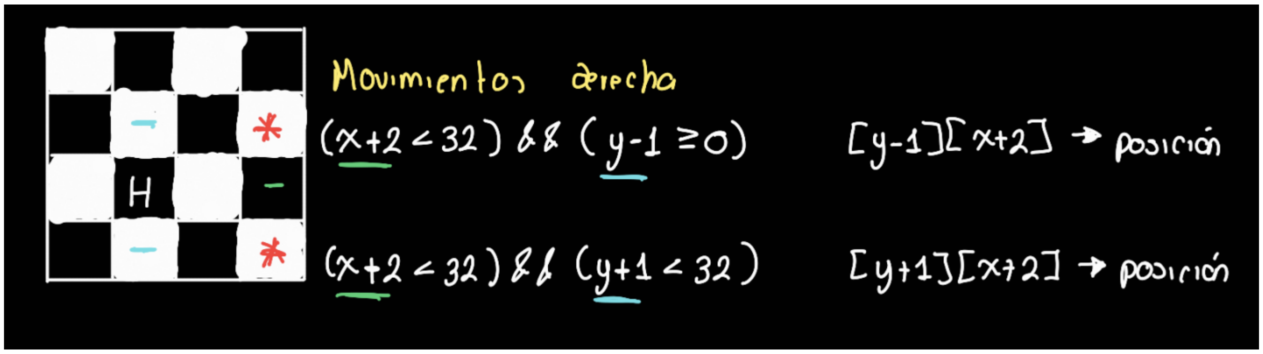


Figure 8 (Horse/knight legal moves).

Pseudocode.

```
Set tablero[y][x] = 'H'
```

```
if (x + 2 < 32) and (y - 1 >= 0) then
    Set tablero[y - 1][x + 2] = 'x'
end if
if (x + 2 < 32) and (y + 1 < 32) then
    Set tablero[y + 1][x + 2] = 'x'
end if

if (x - 2 >= 0) and (y - 1 >= 0) then
    Set tablero[y - 1][x - 2] = 'x'
end if
if (x - 2 >= 0) and (y + 1 < 32) then
    Set tablero[y + 1][x - 2] = 'x'
end if

if (x + 1 < 32) and (y + 2 < 32) then
    Set tablero[y + 2][x + 1] = 'x'
end if
if (x - 1 >= 0) and (y + 2 < 32) then
    Set tablero[y + 2][x - 1] = 'x'
end if

if (x + 1 < 32) and (y - 2 >= 0) then
```

```
    Set tablero[y - 2][x + 1] = 'x'
end if
if (x - 1 >= 0) and (y - 2 >= 0) then
    Set tablero[y - 2][x - 1] = 'x'
end if
```

Tower logic.

Tower moves straight up, down, left and right. We could model this with a for loop to increase only one coordinate.

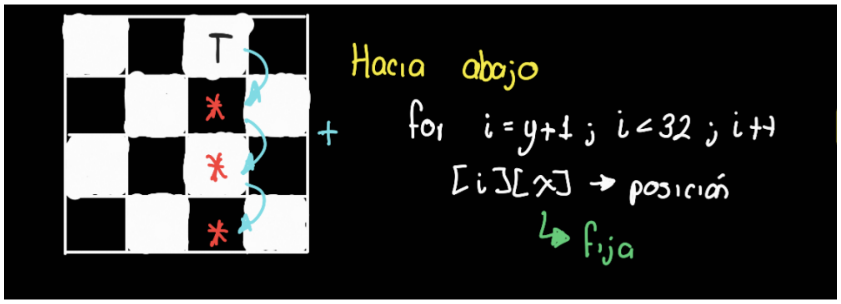


Figure 9 (Tower legal move with the for loop description).

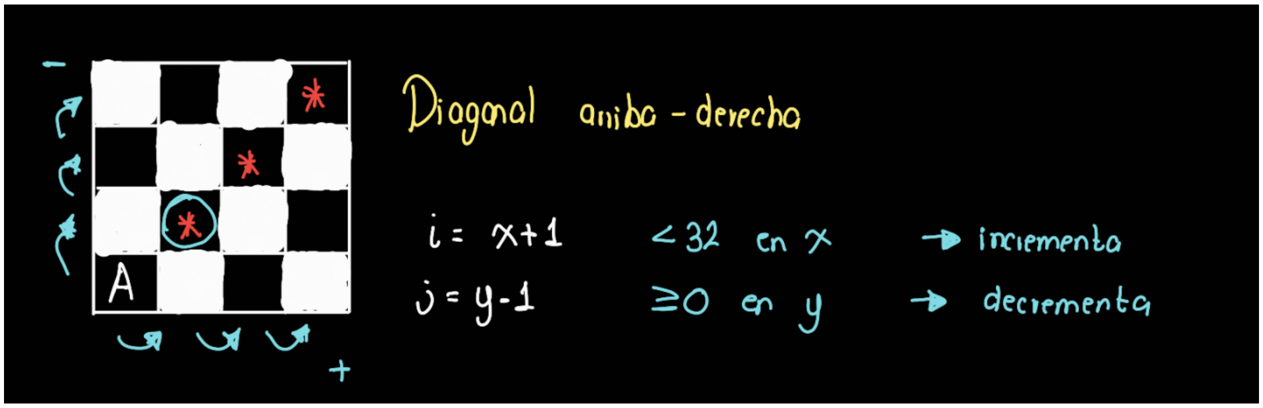
Pseudocode.

```
Set tablero[y][x] = 'T'


for i = x + 1 to 31 do
    Set tablero[y][i] = 'x'
end for


for i = x - 1 downto 0 do
    Set tablero[y][i] = 'x'
end for
```

```
for i = y - 1 downto 0 do
    Set tablero[i][x] = 'x'
end for


for i = y + 1 to 31 do
    Set tablero[i][x] = 'x'
end for
```

Bishop logic.

Bishop moves diagonally, this is telling us that we need to modify both coordinates with a for loop, based in the quadrant of the move.



Diagonal arriba - derecha
i = x+1 < 32 en x → incrementa
j = y-1 ≥ 0 en y → decrementa
A
-
+

Figure 10 (Bishop legal moves).

Pseudocode.

```
Set tablero[y][x] = 'B'
for i = x + 1, j = y - 1; (i < 32) and (j >= 0); i++, j-- do
    Set tablero[j][i] = 'x'
end for
```

```
for i = x - 1, j = y - 1; (i >= 0) and (j >= 0); j--, i-- do
    Set tablero[j][i] = 'x'
end for
for i = x + 1, j = y + 1; (i < 32) and (j < 32); j++, i++ do
    Set tablero[j][i] = 'x'
end for
for i = x - 1, j = y + 1; (i >= 0) and (j < 32); i--, j++ do
    Set tablero[j][i] = 'x'
end for
```

Queen logic.

- We just need to merge tower and bishop’s logics.

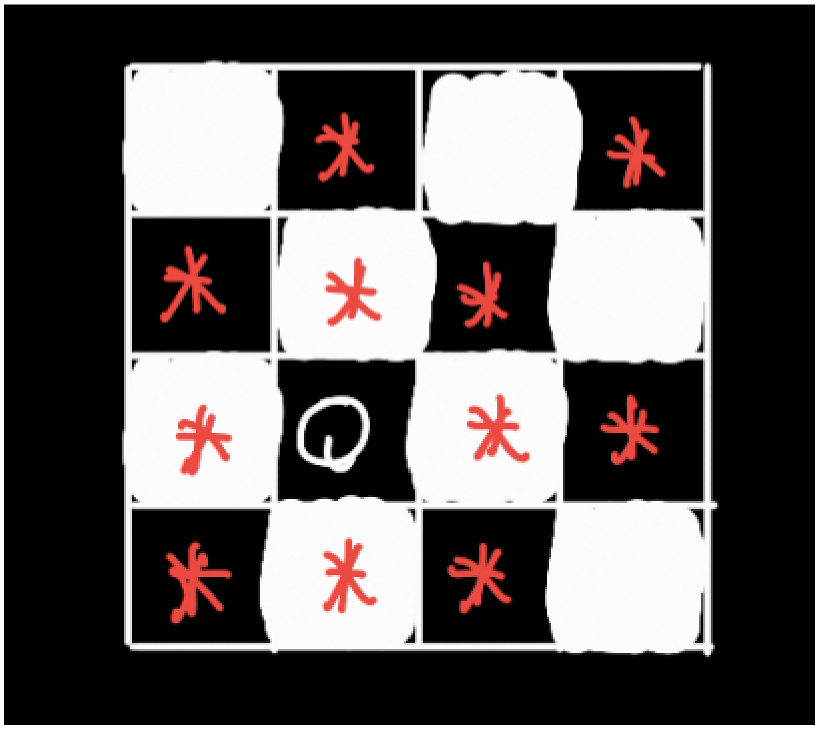


Figure 11 (Queen possible moves).

Pseudocode.

```
Set tablero[y][x] = 'Q'
for i = x + 1, j = y - 1; (i < 32) and (j >= 0); i++, j-- do
   Set tablero[j][i] = 'x'
end for
for i = x - 1, j = y - 1; (i >= 0) and (j >= 0); j--, i-- do
   Set tablero[j][i] = 'x'
end for
for i = x + 1, j = y + 1; (i < 32) and (j < 32); j++, i++ do
   Set tablero[j][i] = 'x'
end for
for i = x - 1, j = y + 1; (i >= 0) and (j < 32); i--, j++ do
   Set tablero[j][i] = 'x'
end for
for i = x + 1; i < 32; i++ do
   Set tablero[y][i] = 'x'
end for
for i = x - 1; i >= 0; i-- do
   Set tablero[y][i] = 'x'
end for
for i = y - 1; i >= 0; i-- do
   Set tablero[i][x] = 'x'
end for
for i = y + 1; i < 32; i++ do
```

```
    Set tablero[i][x] = 'x'
end for
```

Conclusion: The hypothesis was correct, because the abstraction we’ve made since it worked for every piece. There was only a little detail about the pawn, we forgot the attack positions, but during the development we’ve added that move.

I enjoyed this second practice. These kinds of exercises are very helpful for learning the language syntax and starting with the OOP paradigm fundamentals.

References:

- None.