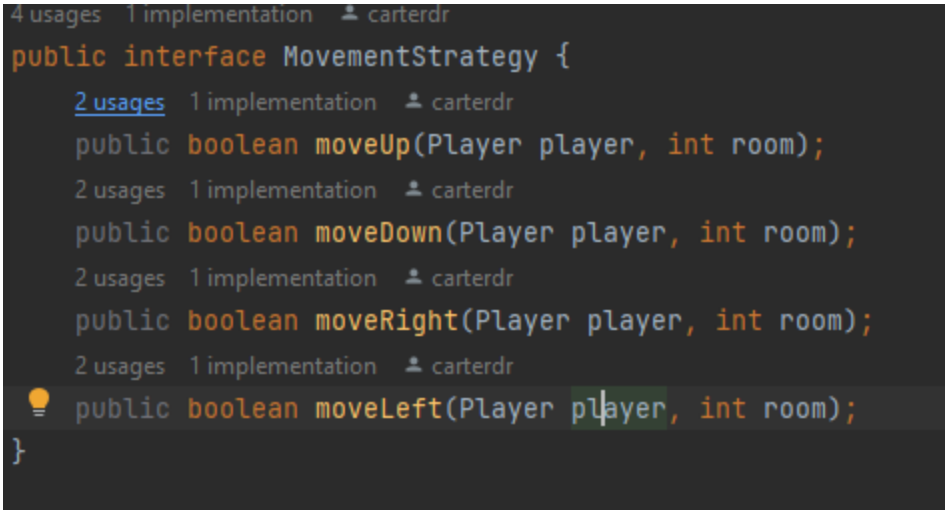
Strategy Pattern



```
4 usages  1 implementation  carterdr
public interface MovementStrategy {
    2 usages  1 implementation  carterdr
    public boolean moveUp(Player player, int room);
    2 usages  1 implementation  carterdr
    public boolean moveDown(Player player, int room);
    2 usages  1 implementation  carterdr
    public boolean moveRight(Player player, int room);
    2 usages  1 implementation  carterdr
    public boolean moveLeft(Player player, int room);
}
```

```
import android.graphics.PointF;
import android.graphics.RectF;
import android.util.Log;
import android.view.KeyEvent;

15 usages  carterdr *
public class WalkMovement implements MovementStrategy {
    8 usages
    private int playerWidth = 130;
    8 usages
    private int playerHeight = 130;
    4 usages
    private int moveAmount = 20;
    2 usages  carterdr
    @Override
    public boolean moveUp(Player player, int room) {...}

    2 usages  carterdr
    @Override
    public boolean moveDown(Player player, int room) {

        float x = player.getPosition().x;
        float y = player.getPosition().y;


        float newX = x;
        float newY = y;

        // Create a RectF to represent the player's current position and dimensions
        // I'm assuming here you have the width and height for the player
        RectF playerRect = new RectF(x, y,  right: x + playerWidth,  bottom: y + playerHeight);

        // Create a new RectF to represent the player's proposed new position
        RectF newPlayerRect = new RectF(playerRect);

        newY = y + moveAmount;
        newPlayerRect.set(x, newY,  right: x + playerWidth,  bottom: newY + playerHeight);

        if (!Wall.isCollision(newPlayerRect, room)) {
            player.setPosition(new PointF(newX, newY));
            Log.d( tag: "Position",  msg: "" + newX + ", " + newY);
            return true;
        }
        return false;
    }
    2 usages  carterdr
    @Override
    public boolean moveRight(Player player, int room) {...}
    2 usages  carterdr *
    @Override
    public boolean moveLeft(Player player, int room) {...}
```

Our code properly uses the strategy pattern when it comes to movement. There is a movementStrategy interface which has the methods up, down, left and right. Then we have implemented 1 concrete movement strategy called walk strategy. A player has a movement strategy and by using the player's movement strategy we can move that player up, down, left or right.