```
private void LetAIMove(int lastX, int lastY) {

    AIMoving = true;
    //记录本次ID
    int currentID = RoundId;
    //获取下一步
    Tuple<int, int> nextMove = runningAI.GetNextAIMove(board, lastX, lastY);
    AIMoving = false;
    //记录下一步,由于耗时可能很长，先判断本局游戏有没有被终止
    if (currentID == RoundId && gameRunning) {
        //通知控制层AI的决策
        SendAIPlayChess(nextMove.Item1, nextMove.Item2);
        PlayChess(Role.AI, nextMove.Item1, nextMove.Item2);
    }

}
```

AbstractAIStrategy

+GetNextAIMove:Tuple<int, int>

+CheckGameOverByPiece:Role

+UserPlayPiece:void

+GameStart:void

+GameForcedEnd:void

BoardManager

-runningAI:AbstractAIStrategy

-LetAIMove(int lastX, int lastY):void

+AskAIMove(int lastX, int lastY):void

```
public void AskAIMove(int lastX, int lastY) {
    LetAIMove(lastX, lastY);
}
```

MCTS	MinMax	MinMaxMCTS	Negmax
略	略	略	略