CS 61B: Code for Lecture 17

Minimax game tree search (no pruning)

```
public class Grid {
  public static final boolean COMPUTER = true;
  public static final boolean HUMAN = false;

  public Best chooseMove(boolean side) {
    Best myBest = new Best();     // My best move
    Best reply;          // Opponent’s best reply

    if ("this" Grid is full or has a win) {
      return a Best with Grid’s score, no move;
    }

    if (side == COMPUTER) {
      myBest.score = -1;
    } else {
      myBest.score = 1;
    }
    myBest.move = any legal move;

    for (each legal move m) {
      perform move m;     // Modifies "this" Grid
      reply = chooseMove(! side);
      undo move m;        // Restores "this" Grid
      if ((side == COMPUTER &&
           reply.score > myBest.score) ||
          (side == HUMAN &&
           reply.score < myBest.score)) {
        myBest.move = m;
        myBest.score = reply.score;
      }
    }

    return myBest;
  }
}

public class Best {
  Move move;
  int score;
}
```

Minimax game tree search with alpha-beta pruning

```
public class Grid {
  public static final boolean COMPUTER = true;
  public static final boolean HUMAN = false;

  public Best chooseMove(boolean side, int alpha, int beta) {
    Best myBest = new Best();       // My best move
    Best reply;            // Opponent’s best reply

    if ("this" Grid is full or has a win) {
      return a Best with the Grid’s score, no move;
    }

    if (side == COMPUTER) {
      myBest.score = alpha;
    } else {
      myBest.score = beta;
    }
    myBest.move = any legal move;

    for (each legal move m) {
      perform move m;       // Modifies "this" Grid
      reply = chooseMove(! side, alpha, beta);
      undo move m;          // Restores "this" Grid
      if ((side == COMPUTER) &&
          (reply.score > myBest.score)) {
        myBest.move = m;
        myBest.score = reply.score;
        alpha = reply.score;
      } else if (side == HUMAN &&
                 reply.score < myBest.score) {
        myBest.move = m;
        myBest.score = reply.score;
        beta = reply.score;
      }

      if (alpha >= beta) {
        return myBest;
      }
    }

    return myBest;
  }

  public Best chooseMove(boolean side) {
    return chooseMove(side, -1, 1);
  }
}
```