```java
/** Game solver searches game tree and finds win value of POSITION.
 *  Caches previously-found win values of POSITION to optimize time.
 *  @author janise
 */
public class Solver {

    Game game;
    String[] cache;

    public Solver(Game g) {
        game = g;
        cache = new String[g.start()+1];
    }

    public String solve(int position) {
        if (cache[position] != null) {
            return cache[position];
        }
        String state = Game.primitiveValue(position);
        if (!state.equals("not_primitive")) {
            return setCache(position, state);
        }
        int[] moves = game.generateMoves(position);
        String[] future = new String[moves.length];
        for (int i = 0; i < moves.length; i++) {
            int newPos = Game.doMove(position, moves[i]);
            future[i] = solve(newPos);
        }
        boolean flag = true;
        for (String f : future) {
            if (f.equals("lose")) {
                return setCache(position, "win");
            } else {
                flag = flag && f.equals("win");
            }
        }
        return setCache(position, flag ? "lose" : "tie");
    }

    String setCache(int pos, String val) {
        cache[pos] = val;
        return val;
    }
}
```

```java
import java.util.ArrayList;

/** Parent class of TenGame and TwentyFiveGame. */
public abstract class Game {
    static int[] legalMoves;
    static int start;

    static int doMove(int position, int move) { return position - move; }

    int[] generateMoves(int position) {
        ArrayList<Integer> moves = new ArrayList<>();
        for (int mv : legalMoves) {
            if (mv <= position) { moves.add(mv); }
        }
        int[] res = new int[moves.size()];
        for (int i = 0; i < res.length; i++) {
            res[i] = moves.get(i);
        }
        return res;
    }

    int start() { return start; }

    static String primitiveValue(int position) {
        return position == 0 ? "lose" : "not_primitive";
    }

    static void solve(Solver s) {
        for (int i = start; i >= 0; i -= 1) {
            System.out.println(String.format("%02d: ", i) + s.solve(i));
        }
    }
}

/** Implements 10-to-0-by-1-or-2 game. Players start with 10 items
 *  and alternate picking 1 or 2 items. The player that takes the
 *  final item wins the game.*/
public class TenGame extends Game {

    public TenGame() {
        legalMoves = new int[]{1, 2};
        start = 10;
    }
    /** Runs solver on TenGame. */
    public static void main(String[] args) {
        Game g = new TenGame();
        Solver s = new Solver(g);
        solve(s);
    }
}
```

```
Output of solving TenGame
--------
10: win
09: lose
08: win
07: win
06: lose
05: win
04: win
03: lose
02: win
01: win
00: lose
```

```java
/** Implementats 25-to-0-by-1-or-3-or-4. Players start with 10 items
 *  and alternate picking 1, 3, or 4 items. The player that takes the
 *  final item wins the game.
public class TwentyFiveGame extends Game {

    public TwentyFiveGame() {
        legalMoves = new int[]{1, 3, 4};
        start = 25;
    }

    public static void main(String[] args) {
        Game g = new TwentyFiveGame();
        Solver s = new Solver(g);
        solve(s);
    }
}
```

```
Output of solving TwentyFiveGame
--------
25: win
24: win
23: lose
22: win
21: lose
20: win
19: win
18: win
17: win
16: lose
15: win
14: lose
13: win
12: win
11: win
10: win
09: lose
08: win
07: lose
06: win
05: win
04: win
03: win
02: lose
01: win
00: lose
```