The Game of Elevenses¹

Timothy J Rolfe
Department of Computer Science
Eastern Washington University
319F CEB
Cheney, WA  99004-2493 USA
Timothy.Rolfe@mail.ewu.edu
http://penguin.ewu.edu/~trolfe/

Abstract

It is becoming increasingly difficult to develop programming assignments for which the students cannot find implementations on-line. This article describes a fairly simple game for which at the moment this is not the case. It is a card game based on scoring face cards and pairs of non-face cards totaling eleven.

General Terms
Algorithms, Design

Keywords
Card game

Categories and Subject Descriptors
F.2.2  Nonnumerical Algorithms and Problems — Computations on discrete structures

Introduction

Blackjack is a common game to implement in an introductory programming class. The problem, however, is that the web is teeming with implementations of blackjack as a text-based game or as a graphical game. Students may be tempted to turn the assignment into web searching rather than programming.

Early one morning, as I was laying in bed staring at the insides of my eyelids and thinking about card games for which this is not true, I thought of a possibility based not on 21 but on 11 and on the ways of combining pairs of cards to total 11.

The Game of Elevenses

Play is modeled on draw poker, but with different evaluation rules. Players are dealt five cards or possibly more. Each player is allowed to replace zero or more cards. Then the hands are

¹ © ACM, (2011). This is the author’s version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Inroads, Vol 2, No. 3, (September 2011) http://doi.acm.org/10.1145/2003616.2003629.
evaluated based on the pairs of number cards that total 11 (that is, 10/ace, 9/2, 8/3, etc.) and face cards, which have the values of 11/J, 12/Q, and 13/K. These are all added together. Thus the highest score possible is 64, a hand of four kings (52) and one queen (12). Non-face cards that are not part of a pair totaling 11 are singletons and do not themselves contribute to the score. Hands having identical scores are ranked based on individual cards. The hand is ordered with face cards in descending order on the left, followed by pairs totaling 11 ranked by their higher card, followed by singletons according to their rank. If two hands have the same score, the rank is determined by comparing the cards from left to right, the first position having cards of differing rank determines the higher hand. If all of the cards have the same rank, then the winning hand is determined by the suit of the left-most card, using the bridge convention: spades before hearts before diamonds before clubs.

The above three hands (all scoring 22) are given in order of their ranking: the first two are identical for the rank of all of the cards, and the suit of the highest card determines the winner, hearts ranking higher than clubs. The second two are not identical for the rank, and in this case the rank of the singleton card determines the winner, five ranking higher than four.

---

2 Images taken from http://hubpages.com/_qii05z0u86du/hub/playing-cards-clip-art, which contains “free playing cards clip art.”
The score of the winning hand is added to that player's running total score. Play continues until
one player has a total score at or above 500. Along the way there are opportunities for additional
side-bets.

Of the many possible, the following is an implementation in Java without the use of user-defined
classes. It could easily be modified to take advantage of user-defined classes.

Each card is represented by a character string beginning either with an alphabetic character
(JQK) for a face card or a one- or two-digit number giving the face value of the card. Note that
aces are low and represented by 1. The string is terminated by an alphabetic character (CDHS)
giving the suit of the card. To get a unique value for each card, generate a two- or three-digit
number: the trailing digit will represent the suit as 0-3, using the same order as is used in bridge
(in increasing order, clubs, diamonds, hearts, and spades). Value/10 gives the rank of the card,
as 1 through 13.

After generating the standard deck as an array of 52 strings, shuffle the deck. Within Java this
can easily be done by taking advantage of the Collections class and the Arrays class. Within the
static class Arrays is the method Arrays.asList(T... a), which returns a fixed-size list backed by
the specified array so that changes to the returned list “write through” to the array. Shuffling the
array is simply a matter of shuffling the list, which is easily done through the utility method
Collections.shuffle (List<?> list). [A quick-and-dirty way to print out an array of objects takes
advantage of Arrays.asList: System.out.println(Arrays.asList(a)).]

During each round of play, cards are dealt to all players. This would be an appropriate time for
initial bets. Then each player has the opportunity to toss in cards (zero or more) and receive
replacements. Here is the time for a final round of bets. Then the winner of the round is
determined as described below.

Each hand is an array of strings representing the cards. Evaluating the hand is greatly facilitated
by ordering the cards thus: give the face cards and the scoring card pairs in decreasing order;
that is, face cards from king down through jack, ordered by suit as the secondary key. Following
this will be pairs of cards that total 11, in the order of larger followed by smaller card, again
ordered by suit of the larger card as the secondary key. The hand ends with the unpaired cards
 singleton s), again ordered by rank and then suit. This arrangement eases the scoring. The total
score for the hand is the sum of the face card values plus the pairs totaling eleven. In case two
players have hands with the same score, the winner will be determined by the rank of the each
card (from 13 down to 1) when hands are compared from highest to lowest rank, the higher rank
winning the hand. [This could even be the ranks of two singleton cards, as for hands 2 and 3 in
the above example.] In case of duplicate hands, the suit of the highest card determines the
winner: in decreasing order, spades, hearts, diamonds, and clubs.
Incremental Development

The game can be developed in three stages: hand evaluation, single game execution, and multi-game tournament.

This game was initially developed for possible inclusion in the Pacific Northwest’s regional contest for the International Collegiate Programming Contest, and the portion selected for inclusion was precisely that of evaluating hands and determining the winners. Consequently the user interface was to accept sets of hands, evaluate them, and report the winner. [The game was not used in the contest.]

The next level is to execute a single game. This means developing the logic to deals hands of cards to players who must be distinguished from each other by game strategies. It might be interesting to challenge the students to devise their own strategies. The author chose to examine the following four strategies:

- Player 0 never exchanges cards after the initial deal.
- Player 1 exchanges all unmatched cards of face value < 8, hoping for a more valuable hand in case of a tied score.
- Player 2 exchanges all unmatched cards of whatever face value.
- Player 3 will even toss in one 11-match hoping for face cards.

Once the play of a single game has been developed, the natural next step is to see which one of the strategies is the winning strategy: develop a program to run a tournament of games, continuing until one of the players has won some specified number of games, ignoring the individual totals by which the player won each of the games.

Web Resource

The author’s programs investigating this game are available through the following URL: http://penguin.ewu.edu/~trolfe/Elevenses/

The description of the problem for possible use in the ICPC regional contest is titled “VanHelsing.doc” — the contest had a horror theme and the description reflects trying to shoe horn the problem into that context. The solution is the program labeled “Elevenses.java”, with input files “11.in” and “judge.in”. The other programs have names that identify them as running either single games or tournaments.