Practical Assignment 1

OO programming in games

February 13, 2012

In this assignment, you will program a simple game using basics C++ and object oriented programming concepts.

Blackjack

The game works as follows (basics rules of play only). Players are dealt cards with point values. Each player tries to reach a total of 21 (called Blackjack) without exceeding that amount. Numbered cards count as their face value. An ace counts as either one or 11 (whichever is best for the player), and any jack, queen or king counts as 10.

The computer is the house and it competes against one to seven players. At the beginning of the round, all participants (including the house) are dealt two cards. The players can see all of the cards, except one of the house's cards. But the house has to reveal directly the second card if it makes its hand a blackjack. Next, each player gets the chance to take one additional card at a time for as long as the player likes (called *hit*). If the player's cards exceed 21, the player loses (*busts*). When all players stop taking additional cards, the house reveals its hidden card and takes automatically additional cards as long as its total is 16 or less. If the house busts, all players who have not busted before win. Otherwise, the player wins if its total is greater than the house's. If totals are the same, it is a tie (*push*) (Fig. 1).

- The use of the following class hierarchy is advised, small adjustments are allowed if justified (by comments in the code).

Class	**Base class**	**Description**
`Card`	None	A playing card
`Hand`	None	A collection of `Card` objects
`Deck`	`Hand`	A `Hand` that has extra functionalities such as shuffling and dealing
`GenericPlayer`	`Hand`	A generic player is mostly represented by its hand
`Player`	`GenericPlayer`	A human player
`House`	`GenericPlayer`	The computer player, i.e. the house
`Game`	None	A game

- The header files of `Hand`, `GenericPlayer` and `Deck` and the `main` file are given in Appendix.
- Write the game with only public member functions and protected data members.
- Each instance of `Card` is unique. Do not create several instances of the same card. When moving cards from the deck to the player's hand, move pointers, do not copy objects. A hand thus contains a vector of pointers to `Card` objects.
- A player can be seen as a hand with a name.
- The deck will deal cards to generic players. A `Deck` object will so have a member function to deal cards that is polymorphic and will work with either a `Player` and a `House`.
- The game contains a deck of cards, the house and several players.

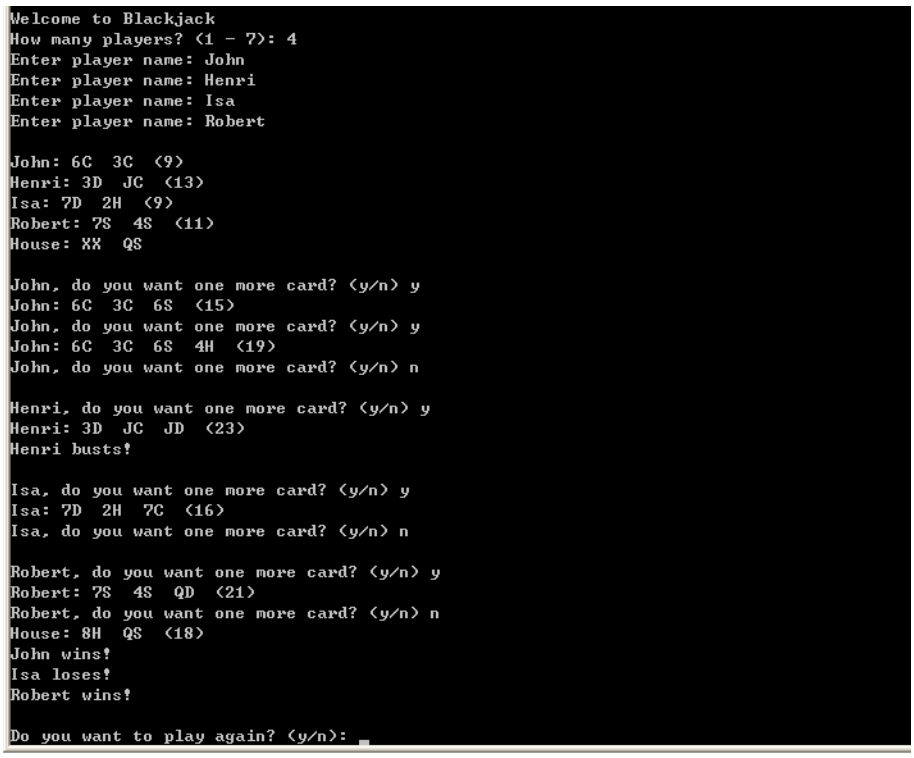

Figure 1: The blackjack game.

- You can shuffle a vector of objects using the `random_shuffle` function of the `algorithm` library.
- You will need among others:
  - an enumeration type describing the rank of a card
  - an enumeration type describing the kind (suit) of a card
  - indications that a card is visible or not and that a player is busted
  - functions to flip a card, add a card to a hand, clear a hand, get the value of a hand, shuffle deck, deal cards, etc.
- The pseudo-code of the game loop (`Game::play()`) is:

Deal players and the house two cards
Hide the house's first card if not Blackjack
Display players' and house's hands
Deal additional cards to players
Reveal house's first card
Deal additional cards to house
If house is busted
    Everyone who is not busted wins
Else
    For each player
        If player is not busted
            If player's hand > house's hand
                Player wins
            Else if player's hand < house's hand
                Player loses

Else
Tie game
Else Player loses
Move everyone's cards back into deck

Submitting your work

The submission deadline is Sunday February 19 at 11:59pm. To send us your work, you will use the CS-UU submission system available at `http://www.cs.uu.nl/docs/submit`. Please DO NOT send your assignment by email.

Design your code in an object-oriented fashion. So, use classes and inheritance where appropriate. Please include sufficient comments to the code so that at least the role of each function and class in your program is clear. Include as much error checking as appropriate. If your project does not compile and run in both debug and release mode, there will be no grading.

Please make sure to *clean* your solution (no temporary files and no .sdf). The assignment consists of the solution file (.sln), the project files (.vcxproj) and the source files (.cpp and .h). Create a .zip or .rar archive of your assignment files and upload it using the submission system. After a successful submission you should receive a confirmation email in your student email account. If not, please contact Jeroen Fokker.

Appendix

Main

```
#include "Game.h"

using namespace std;

int main(){

        cout << "Welcome to Blackjack" << endl;

        // Number of players
        int numPlayers = 0;
        while (numPlayers < 1 || numPlayers > 7) {
                cout << "How many players? (1 - 7): ";
                cin >> numPlayers;
        }

        // Player names
        vector<string> names;
        string name;
        for (int i=0; i<numPlayers; i++) {
                cout << "Enter player name: ";
                cin >> name;
                names.push_back(name);
        }
        cout << endl;

        // Play the game
        Game game(names);
        char again = 'y';
        while (again == 'y' || again == 'Y') {
                game.play();
```

```
            cout << endl << "Do you want to play again? (y/n): ";
            cin >> again;
        }

        system("pause");
        return 0;
}
```

Hand header

```
#ifndef HAND_HEADER
#define HAND_HEADER

#include <vector>
#include <string>
#include <iostream>

#include "Card.h"

class Hand {

public:

        // constructor and destructor
        Hand();
        virtual ~Hand();

        // adds a card to the hand
        void add(Card *);

        // clears hand of all cards
        void clear();

        // get hand value
        int getTotal() const;

protected:

        std::vector<Card *> cards_;
};

#endif
```

GenericPlayer header

```
#ifndef GENERICPLAYER_HEADER
#define GENERICPLAYER_HEADER

#include "Hand.h"
#include <string>
#include <iostream>

class GenericPlayer : public Hand {

public:
```

```
        // constructor and destructor
        GenericPlayer(const std::string& name = "");
        virtual ~GenericPlayer();

        // indicates if player wants to hit
        virtual bool isHitting() const = 0;

        // is player busted
        bool isBusted() const;

        // annouces that player busts
        void bust() const;

        // print the hand with name
        virtual void printHand() const = 0;

protected:
        std::string name_;

};

#endif
```

Deck header

```
#ifndef DECK_HEADER
#define DECK_HEADER

#include <string>
#include <iostream>
#include <algorithm>

#include "Hand.h"
#include "GenericPlayer.h"

class Deck : public Hand {

public:

        // constructor and destructor
        Deck();
        virtual ~Deck();

        // create the deck of 52 cards
        void populate();

        // shuffle cards
        void shuffle();

        // deal one card to a hand
        void dealOneCard(Hand&);

        // give cards to generic player as long as not busted and asks for it
        void giveCards(GenericPlayer&);

};

#endif
```