Documentation

1. *Description of the game*

Klondike is a patience game (solitaire card game). In the U.S. and Canada, Klondike is the best-known solitaire card game, to the point that the term "solitaire", in the absence of additional qualifiers, typically refers to Klondike.
The game is still extremely popular and thus many know its rules. The card deck consists of 52 cards. The deck is shuffled and divided into 4 tables: the main table consists of 7 columns indexed from left to write as 1 to 7, with each column containing that many cards as its index and only the last card is flipped. The rest of the cards form the stock which is placed in the upper left corner of the layout, un-flipped.
In the rightmost corner there are 4 empty spaces, called the foundation, that should be built up in ascending order from Ace to King of the same suit/type (diamonds, hearts, spades, clubs).
Every face-up card in a partial pile, or a complete pile, can be moved, as a unit, to another tableau pile on the basis of their highest card. Any empty piles can be filled with a King, or a pile of cards with a King. The aim of the game is to build up four stacks of cards starting with Ace and ending with King, all of the same suit, on one of the four foundations, at which time the player would have won. There are different ways of dealing the remainder of the deck from the stock to the waste, including the following:

- Turning three cards at once to the waste, with no limit on passes through the deck.
- Turning three cards at once to the waste, with three passes through the deck.
- Turning one card at a time to the waste, with three passes through the deck.
- Turning one card at a time to the waste with only a single pass through the deck and playing it if possible.
- Turning one card at a time to the waste, with no limit on passes through the deck.

If the player can not make any moves, the game is considered lost! (Klondike (solitaire) - Wikipedia, n.d.)

2. *Purpose of the project*

The scope of this project was to further understand the nature of complex data structures, how to implement them in an object-oriented program like C++ (list, linked list, double-linked list, vectors in C++). I chose to do this project, because I wanted to challenge myself, to put myself in the situation where I needed to work with abstract and complex data types/structures and to recap/reevaluate/renew my knowledge in C++. The most challenging part of the project was the

KOKOVICS EMANUEL-ATTILA
WEST UNIVERSITY OF TIMISOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
IE / YEAR 2 / GROUP 2
INDIVIDUAL PROJECT

implementation of the classes, because OOP is not my strong suit, even though I understand very well the concept, it sometimes becomes difficult in applying them.

3. *Benefits*

For myself, I can say that it strongly improved my skills and I will actually continue to develop on this project, and construct other projects using the best that C++ has to offers, even though it is a fairly difficult program.

I was in the situation, where I had to research a lot about C++ syntax, how objects of different classes can interact, how inheritance works in this program, what has been added in the last standards and I had to solve different types of errors that arrive even from the most minor choices of implementation made.

4. *Technical description of the program*

I structured the application in headers and cpp file:

a) PlayerCard.h, TableauPlayerCard.h, StackDeckCards.h, Klondike.h
b) PlayerCard.cpp, TableauPlayerCard.cpp, StackDeckCards.cpp, Klondike.cpp
c) main.cpp

The mentioned files can be found in the include folder and src folder, respectively.

I commented my code, because I think it is one of the most important part of the process, because it will aid in easily understanding the code by others, but even by myself when I wanna revisit my old programs.

I initially used Visual Studio Code and also tried Visual Studio, which made me understand better what is happening at preprocessing and compiling time. And I ended up using Code::Blocks, because I could not waste more time, for now, to manage the problems that appeared in Visual Studio Code.

I will briefly discuss each class and its methods.

The PlayerCard class:

- The PlayerCard class contains standard get functions, the constructor is used here to initialize instance variables representing the main properties of the created object.
- In the PlayerCard.cpp file one can see the implementation of each method of PlayerCard class, initially each card is going to be un-flipped and the child and parent pointer are NULL.
- The method getKlondikeValue() returns an integer and its only purpose is to facilitate the implementation of the game rules,.

TableauPlayerCard

- - **actualMaximumAllowedSize: int**
- - **actualSize: int**
- - **playerCards: vector<PlayerCard*>**

- + TableauPlayerCard()
- + ~TableauPlayerCard()
- + TableauPlayerCard(int)
- + operator[](int): PlayerCard&
- + topPlayerCard(): PlayerCard&
- + pushCardOnStack(PlayerCard &): bool
- + pushValueOfCopyCard(PlayerCard card): bool
- + emptyTableau(): bool
- + removePlayerCardAtIndex(int index): bool
- + popLastCard(): bool
- + theSize: int
- + **clearTableauOfPlayerCards(): void**
- + **moveBetween(TableauPlayerCard &, TableauPlayerCard &): static void**

Klondike

- - theGameFullDeck : TableauPlayerCard
- - wastedGameCards : TableauPlayerCard
- - theTableauPlayerCards : vector<TableauPlayerCard>
- - typeOfCard : vector<TableauPlayerCard>

- + Klondike()
- + ~ Klondike ()
- + colonizeMainTableau (): void
- + dealFromStackDeck(): void
- + printTheGame (): void
- + printTheTypeOfCards ():void
- + printTheStackDeck (): void
- + theMoveToFoundation(int): void
- + theMoveBetweenRows(int , int): void
- + theMoveRowToRow(int , int): void
- + theMoveFromStackDeckToRow(int): void
- + validGenericMove(int , int): bool
- + validSpecificRowToRowMove(PlayerCard *, int): bool
- + validSpecificToFoundationMove(int from): bool
- + finishedGame(): bool

PlayerCard

- - cardNumber: char
- - cardType: char
- - isCardFaceUp: bool
- + parent: PlayerCard*
- + child: PlayerCard*

- + PlayerCard()
- + ~PlayerCard()
- + PlayerCard(char, char)
- + getIsCardFaceUp(): bool
- + getCardNumber(): char
- + getCardType(): char
- + getKlondikeValue(): int
- + ostream& operator<< (std::ostream & , PlayerCard&)
- + Flip(): void

StackDeckCard

- - someRandomDeck: PlayerCard[ ]
- - currentIndex: int

- + StackDeckCards()
- + ~ StackDeckCards()
- + colonizeStackDeck(): void
- + displayStackDeck(): void
- + colonizeVector(TableauPlayerCard &): void
- + shuffleStackDeck(): void

KOKOVICS EMANUEL-ATTILA
WEST UNIVERSITY OF TIMISOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
IE / YEAR 2 / GROUP 2
INDIVIDUAL PROJECT

The TableauPlayerCard class:

- These class is to aid in creating and manipulating list/vectors of different sizes.
- The important field in this class is: std::vector<PlayerCard*> playerCards;
- The custom constructor here is also a basic one, used for initializing variables.
- pushCardOnStack(PlayerCard &card) method is going to be used to verify if there is still space on a column of card, and if so the card will be added on top of that stack.
- In this game you are allowed to move card only when certain criteria is met, thus a method like topPlayerCard() that returns a reference to a top card will be very usefull later.
- popLastCard() removes the last card on a column.
- Thinking about the game rules, all the methods of these class are natural and straight forward.

The StackDeckCards class:

- These class make use of the other two classes.
- In the header file one can see 4 global constant static variables, one for the number of ranks (13 in total), one for the suits (4 in total: diamonds, hearts, clubs and spades), and the remaining two contain the actual symbols used in this game.
- As private fields, we have two, one of which is most important, an array of a fixed size of 52 and type PlayerCard, because in fact that is how many card the deck for the game has to have.
- Four methods are present, they are in this class to aid in the construction of the deck, in displaying the deck cards and in shuffling the deck; the custom constructor actually make a call to the method that aid in populating the deck;
- The shuffleStackDeck uses a pseudo-random method, that is sufficiently good for the purpose of this project.

The Klondike class:

- Makes use of the class StackDeckCards, as one can see in its constructor; in fact it creates the same deck, but because immediately after it calls the suffleStackDeck() method, we obtaind each time another arrangement of the cards.
- In its header file we see four fields: two simple ones of type TableauPlayerCard, and the other two are vectors containing variables of type TableauPlayerCard ().
- Now that we have the deck, we can construct the main table (which have 7 columns, each with a specific number of cards, in fact we only set aside enough

size for it and after that we call the method that fills it) and the foundation table (just set aside the exact size it needs, but we leave it empty).

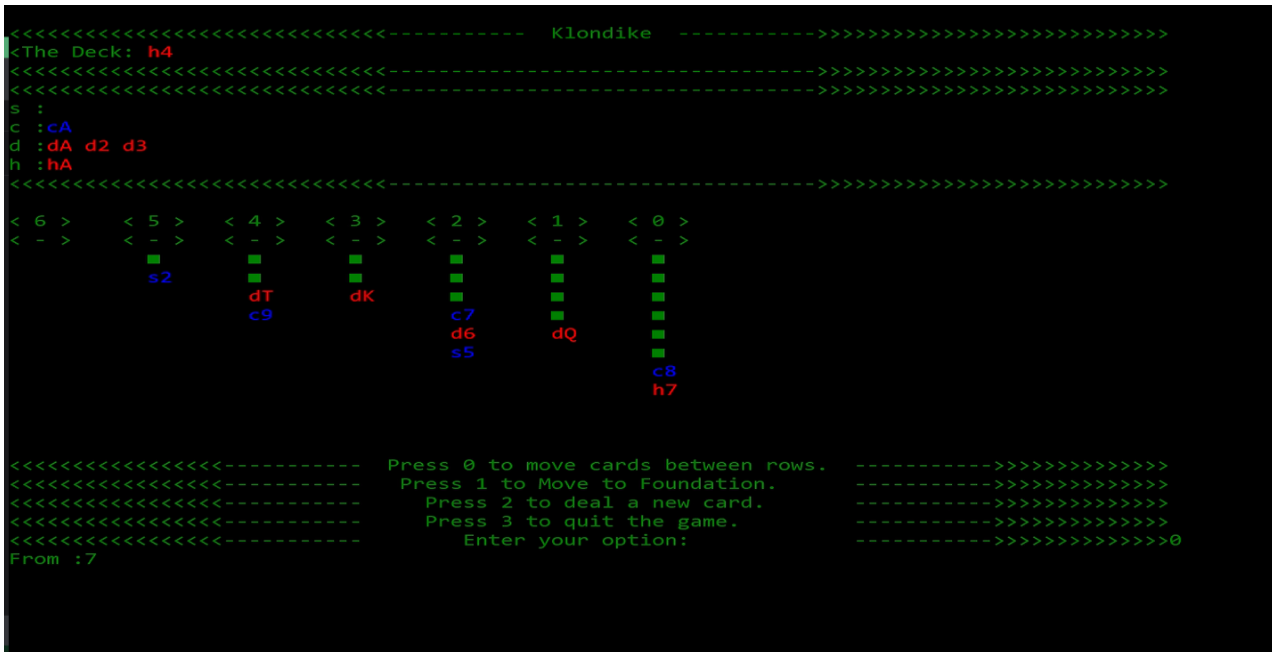
- The rest of the methods are there to validate a player move, and if possible make the move.
- The methods take into consideration every rule: for example on an empty column we can only move a KING, and we can move card based on their type(color) and number.
- finishedGame() just verifies if the foundation block are full, and if so that means the game has ended successfully.

The main program:

- contains three functions, one of which is used to constantly updating the screen, the other two are aiding in the interactivity of the player with the application.
- The main functions call them and also display the menu options for the player.

Future work:

- Try to improve this project, by trying to use new implemented methods in C++, templates, maps, etc;
- Construct on this project, other similar game cards, and expand this application so that the user can choose from more than 2 games;
- Definitely try to make a similar project that will use graphics;
- Develop a simple graphical game in C++ using SDL2 library.



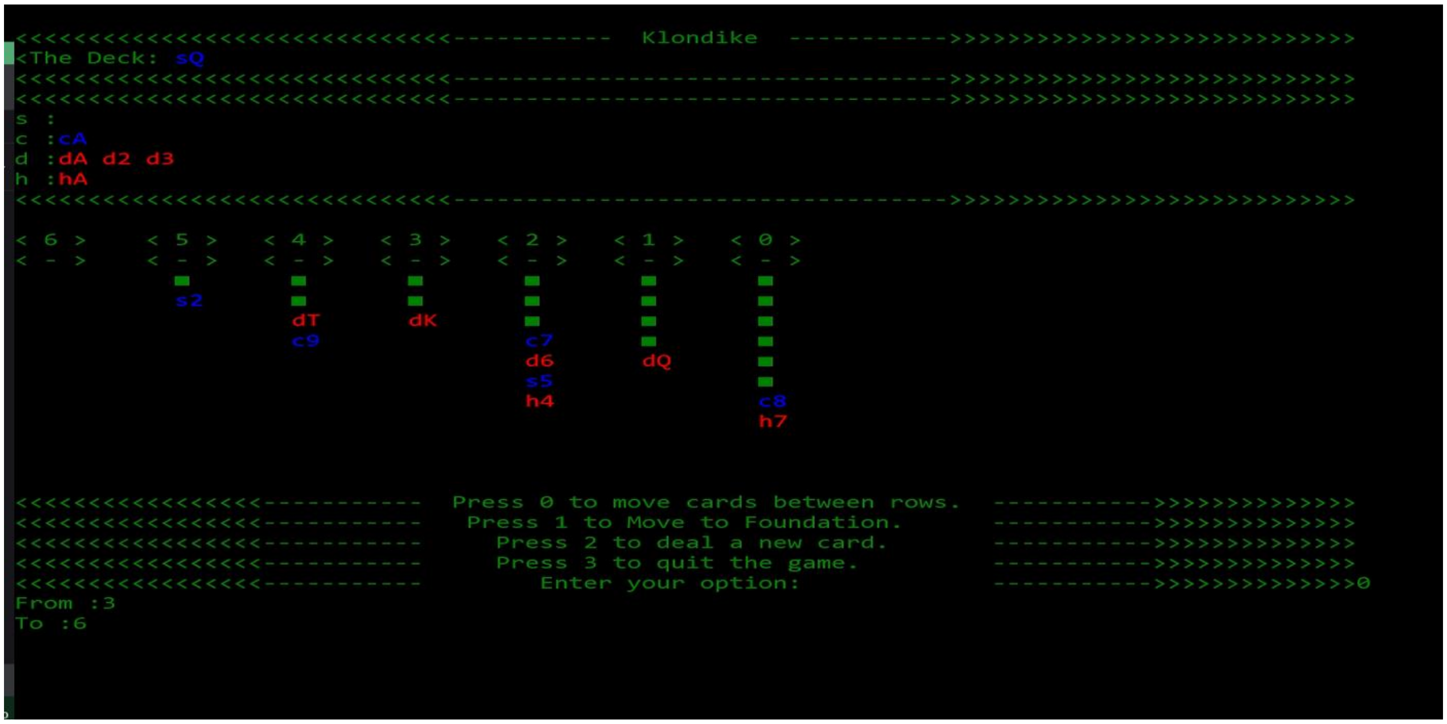
```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<-----------  Klondike  ----------->>>>>>>>>>>>>>>>>>>>>>>>>>>>
<The Deck: h4
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<---------------------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<---------------------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>
s :
c :cA
d :dA d2 d3
h :hA
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<---------------------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>

< 6 >    < 5 >   < 4 >   < 3 >   < 2 >   < 1 >   < 0 >
< - >    < - >   < - >   < - >   < - >   < - >   < - >
           ■       ■       ■       ■       ■       ■
           s2      ■       ■       ■       ■       ■
                   dT      dK      ■       ■       ■
                   c9              c7      ■       ■
                                   d6      dQ      ■
                                   s5              ■
                                                   c8
                                                   h7



<<<<<<<<<<<<<<<<<-----------  Press 0 to move cards between rows.  ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------   Press 1 to Move to Foundation.      ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------     Press 2 to deal a new card.       ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------     Press 3 to quit the game.         ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------        Enter your option:             ----------->>>>>>>>>>>>>>0
From :7
```

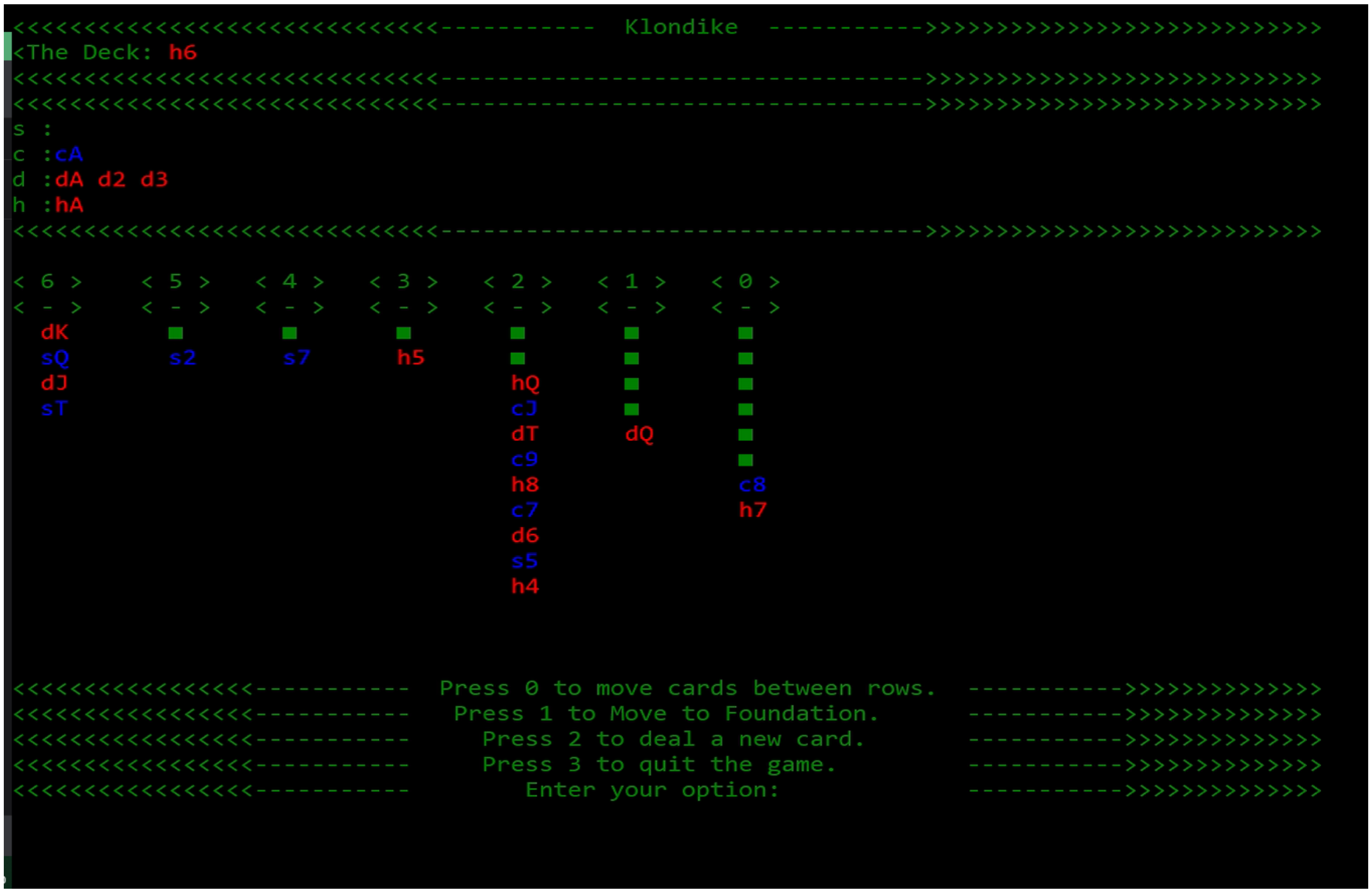
KOKOVICS EMANUEL-ATTILA
WEST UNIVERSITY OF TIMISOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
IE / YEAR 2 / GROUP 2
INDIVIDUAL PROJECT



```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<------------  Klondike  ----------->>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<The Deck: sQ
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<---------------------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<---------------------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
s :
c :cA
d :dA d2 d3
h :hA
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<---------------------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

< 6 >    < 5 >   < 4 >   < 3 >   < 2 >   < 1 >   < 0 >
< - >    < - >   < - >   < - >   < - >   < - >   < - >
           ■       ■       ■       ■       ■       ■
           s2      ■       ■       ■       ■       ■
                   dT      dK      ■       ■       ■
                   c9              c7      ■       ■
                                   d6      dQ      ■
                                   s5              ■
                                   h4              c8
                                                   h7


<<<<<<<<<<<<<<<<<-----------   Press 0 to move cards between rows.   ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------    Press 1 to Move to Foundation.       ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------      Press 2 to deal a new card.        ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------      Press 3 to quit the game.          ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------         Enter your option:              ----------->>>>>>>>>>>>>>0
From :3
To :6
```



```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<------------  Klondike  ----------->>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<The Deck: h6
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<---------------------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<---------------------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
s :
c :cA
d :dA d2 d3
h :hA
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<---------------------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

< 6 >    < 5 >   < 4 >   < 3 >   < 2 >   < 1 >   < 0 >
< - >    < - >   < - >   < - >   < - >   < - >   < - >
  dK       ■       ■       ■       ■       ■       ■
  sQ       s2      s7      h5      ■       ■       ■
  dJ                               hQ      ■       ■
  sT                               cJ      ■       ■
                                   dT      dQ      ■
                                   c9              ■
                                   h8              c8
                                   c7              h7
                                   d6
                                   s5
                                   h4


<<<<<<<<<<<<<<<<<-----------   Press 0 to move cards between rows.   ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------    Press 1 to Move to Foundation.       ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------      Press 2 to deal a new card.        ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------      Press 3 to quit the game.          ----------->>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<-----------         Enter your option:              ----------->>>>>>>>>>>>>>
```

KOKOVICS EMANUEL-ATTILA
WEST UNIVERSITY OF TIMISOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
IE / YEAR 2 / GROUP 2
INDIVIDUAL PROJECT

References

1) https://atomicobject.com/resources/oo-programming/other-oo-class-relationships

2) https://www.w3schools.in/c-tutorial/c-header-files/

3) https://www.geeksforgeeks.org/vector-in-cpp-stl/

4) http://www.cplusplus.com/reference/vector/vector/

5) https://www.edureka.co/blog/vectors-in-cpp/

6) https://www.tutorialspoint.com/cplusplus/cpp_overloading.htm

7) http://www.cplusplus.com/reference/cstdlib/srand/

8) https://www.daniweb.com/programming/software-development/threads/266484/how-to-print-text-into-a-different-color-in-terminal

9) https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/

10) https://en.wikipedia.org/wiki/Klondike_(solitaire)#cite_note-3

11) https://www.solitr.com/klondike-turn-one

12) https://www.learncpp.com/cpp-tutorial/93-overloading-the-io-operators/

13) https://docs.microsoft.com/en-us/windows/console/getstdhandle

14) https://stackoverflow.com/questions/4053837/colorizing-text-in-the-console-with-c