Progetto Ingegneria del Software

Anno Accademico 2017-2018

Sessione estiva

BriscolaX2

Il più famoso gioco di carte cinquecentesco.

Angelo Sasso

Matricola n°263143

Indice

1. Specifica del problema

Il mazzo è composto da 40 carte italiane.
Regole
I punti disponibili per ogni gioco sono in totale 120, vince chi ne realizza almeno 61; se i punti sono 60 per entrambi i giocatori o coppie la partita è pareggiata.
I valori di presa sono nell'ordine decrescente: Asso, 3, Re, Cavallo, Donna o Fante, 7, 6, 5, 4 e 2.
Nella versione proposta potranno giocare due giocatori in una partita uno contro uno condividendo risorse e schermo.

Ogni giocatore ha una mano composta da 3 carte, mentre la prima del mazzo, viene scoperta e posta sul tavolo per designare il seme di briscola; per convenzione la carta che designa la briscola, viene messa sotto il mazzo per tutta la durata della partita, di conseguenza sarà sempre visibile ai giocatori e sarà l'ultima carta ad essere pescata, iniziano quindi una serie di mani.
Ogni giocatore giocherà la sua, ovvero la lancerà sul tavolo, con lo scopo di aggiudicarsi la mano o di totalizzare il maggior numero di punti. Da parte dei giocatori non esiste alcun obbligo di giocare un particolare tipo di seme.
L'aggiudicazione della mano avviene secondo regole molto semplici:
il primo giocatore di mano, così viene chiamato colui che posa la prima carta a ogni giro, determina il seme di mano o dominante e virtualmente è il vincitore temporaneo della mano.
La mano può essere temporaneamente aggiudicata da un altro giocatore se posa una carta del seme di mano, ma con valore di presa maggiore, oppure giocando una qualsiasi carta del seme di briscola, anche con valore di presa inferiore.
La mano viene aggiudicata dal giocatore che ha posato la carta di briscola col valore di presa maggiore, o, in mancanza, da colui che ha giocato la carta del seme di mano comunque col valore di presa maggiore; e infine, in mancanza di carte di briscola o altre carte del seme di mano, dal primo giocatore della mano, in quanto rimane l'unico ad aver giocato la carta del seme di mano.

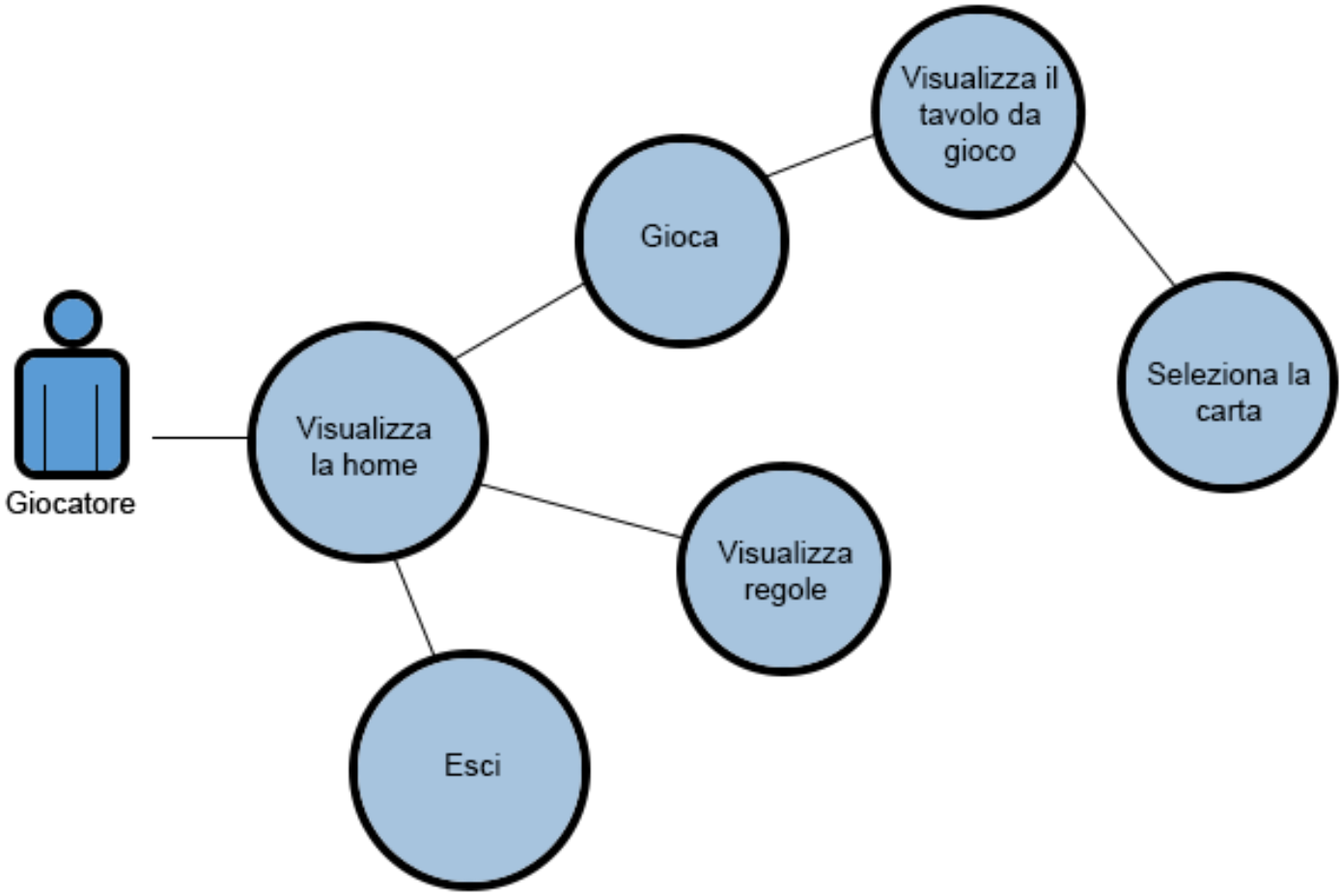
Si badi che il seme di mano, essendo determinato di volta in volta dalla prima carta giocata nella mano, può anche incidentalmente coincidere col seme di briscola; in questo caso la mano se la aggiudica colui che ha giocato la briscola maggiore.
Il vincitore della mano successivamente sarà il primo a prendere la prima carta del tallone, seguito dall'altro giocatore e sarà sempre il primo ad aprire la mano successiva, e quindi a decidere il nuovo seme di mano.
Di seguito sono riportati i valori delle carte in una tabella.

Carta	**Punti**
Asso (detto Carico, termine usato anche per il Tre)	11
Tre (detto Carico, termine usato anche per l'Asso)	10
Re o 10	4
Cavallo o 9	3
Fante o Donna o 8	3
7, 6, 5, 4 e 2 (dette scartine o scartini o lisci)	0

2. Specifica dei requisiti

Di seguito vengono riportate i casi d'usi dell'applicativo che ci aiuteranno a comprendere al meglio le funzionalità del gioco.
Si tratta di un diagramma che esprime un comportamento sulla base dei suoi risultati osservabili. Ognuno dei due giocatori appena è il suo turno, può scegliere di tirare una delle proprie carte. Il vincitore della mano prenderà in automatico una carta dal mazzo prima del perdente, fino a quando il mazzo sarà esaurito.

2.1 Diagramma dei casi d'uso



2.2 Relazioni e descrizione dei casi d'uso

Si mostreranno di seguito le descrizioni dei casi d'uso individuati.

- Il giocatore avvia l'applicazione GUI e visualizzerà un form da cui avrà modo di visualizzare le regole del gioco e di iniziare a giocare. I casi d'uso #1 e #2 sono del tutto speculari.

Caso d'uso	Aprire il gioco
ID	#1
Attore	Giocatore
Precondizioni	L'utente deve aver lanciato l'eseguibile
Corso d'azione base	Il gioco si avvia
Post condizioni	Viene mostrata la home del gioco
Percorso Alternativo	/

Caso d'uso	Scegliere il pulsante "Inizia partita"
ID	#2
Attore	Giocatore
Precondizioni	Il gioco è stato avviato
Corso d'azione base	Viene creata una nuova partita

Post condizioni	Il giocatore si trova nel form della partita
Percorso Alternativo	Se la partita è terminata, la post condizione è valida anche in caso di “Rivincita”

Caso d’uso	Selezionare una carta da lanciare
ID	#3
Attore	Giocatore
Precondizioni	Deve essere il turno del giocatore
Corso d’azione base	1) Il giocatore clicca su una delle carte della mano 2) La carta viene lanciata sul tavolo
Post condizioni	Se il giocatore è il primo a tirare, il turno viene invertito, altrimenti si procede con il confronto
Percorso Alternativo	/

Caso d’uso	Vittoria
ID	#4
Attore	Giocatore
Precondizioni	Avere più punti dell’avversario a fine partita
Corso d’azione base	Viene mostrato un form che annuncia la vittoria
Post condizioni	Si può scegliere se avviare una rivincita o chiudere il gioco

Percorso Alternativo	/

Caso d'uso	Pareggio
ID	#5
Attore	Giocatore
Precondizioni	I giocatori si contendono lo stesso punteggio
Corso d'azione base	Viene mostrato un form annunciante il pareggio
Post condizioni	Viene chiesta un'eventuale rivincita o la terminazione del gioco
Percorso Alternativo	/

Caso d'uso	Regole del gioco
ID	#6
Attore	Giocatore
Precondizioni	Avere aperto il gioco
Corso d'azione base	Viene mostrato un gioco con le regole del gioco
Post condizioni	/
Percorso Alternativo	/

Caso d'uso	Il giocatore lancia una carta di tipo CarteLisci
ID	#7
Attore	Giocatore
Precondizioni	Avere una carta di tipo CarteLisci in mano
Corso d'azione base	Se è la prima carta ad essere giocata durante la mano corrente, il turno viene invertito
Post condizioni	Se è la seconda carta ad essere giocata, si avvia il confronto
Percorso Alternativo	/

Caso d'uso	Confronto carte
ID	#8
Attore	Giocatore
Precondizioni	Il giocatore che ha tirato la carta è il secondo a giocare la mano
Corso d'azione base	Confronta il seme delle due carte
Post condizioni	Vengono assegnati i punti
Percorso Alternativo	/

Caso d'uso	Esci
ID	#9
Attore	Giocatore
Precondizioni	Avere aperto il gioco
Corso d'azione base	Premere il pulsante "Esci"
Post condizioni	Il gioco viene terminato
Percorso Alternativo	Concludere una partita

3. Analisi e progettazione

Descriveremo in questa sezione le principali scelte di progettazione e di implementazione fatte per realizzare il gioco.

Scelte di progetto

- Linguaggio di programmazione
- Ambiente di sviluppo
- GUI
- Design pattern
- Organizzazione progetto
- Classi
- Diagramma delle classi
- Diagramma di interazioni

3.1 Linguaggio di programmazione : C#

Il linguaggio di programmazione C# è un linguaggio multi-paradigma: orientato ad oggetti, imperativo, ad eventi e funzionale. Il linguaggio è stato concepito dalla Microsoft per contrapporsi ai nascenti linguaggi ad oggetto. La gran parte della forza di questo linguaggio deriva dall'utilizzo delle API del framework .NET che mette a disposizione numerose classi base per lo sviluppo di applicazioni di vario genere.
C# risulta essere molto versatile e permette allo sviluppatore di realizzare progetti software sfruttando a pieno le caratteristiche e i vantaggi della programmazione ad oggetti.
In particolare dando la possibilità di effettuare l'incapsulamento dei vari componenti per riuscire a gestire i malfunzionamenti, decidendo il grado di accessibilità dei vari elementi, sfruttando la definizione di classe astratte, interfacce che ci consentono di creare dei format per gli oggetti ben definiti che possono essere ereditati tra le varie classi e sfruttando in ultimo il concetto di polimorfismo che ci consente di avere codice riusabile e più facilmente mantenibile per successive migliorie.
Offre un ottimo ambiente per lo sviluppo visuale delle applicazioni di tipo desktop ed è reperibile sul web un'ampia documentazione delle classi a disposizione.

3.2 Ambiente di sviluppo

Si è utilizzato per lo sviluppo del progetto l'ambiente **Visual Studio 2017 Community** che utilizza il framework .NET 4.0/4.5.2 La versione di Windows utilizzata è Windows 10 Creator Update. Le funzionalità di questo ambiente sono state sufficienti per lo sviluppo del software.

3.3 GUI

Oltre allo sviluppo delle classi riguardanti la logica delle funzionalità del gioco e una funzione main che sfruttasse queste per testare il software al fine di effettuare un debug tramite terminale, si realizza un'applicazione di tipo grafica utilizzando i Windows Form così da rendere realistica la partita. Per le carte è stata utilizzata la classe PictureBox che rappresenta un controllo casella immagine di Windows per la visualizzazione dell'immagine. Le immagini utilizzate dall'applicazione risiedono nei Resources del progetto. Per segnare i punti sono stati utilizzati invece dei label che tengono traccia del cambiamento della variabile punteggio dopo ogni azione fatta dal giocatore. er altri controlli o per spostarsi dal form rappresentante la Home all'effettivo inizio della partita vengono utilizzate altre componenti come i button, che rispondono all'evento click.

Tra un turno e l'altro, quindi ad ogni mossa fatta da ogni giocatore, è inserito anche un MessageBox che delimita l'attesa del prossimo giocatore così da evitare la visione delle carte dell' altro giocatore durante il cambio del turno.

I form introdotti durante lo sviluppo sono i seguenti:

1) Home.
2) View.
3) Fine.

1)Il primo Form introdotto è Home e rappresenta la pagina iniziale che appare all'utente all'apertura del programma.

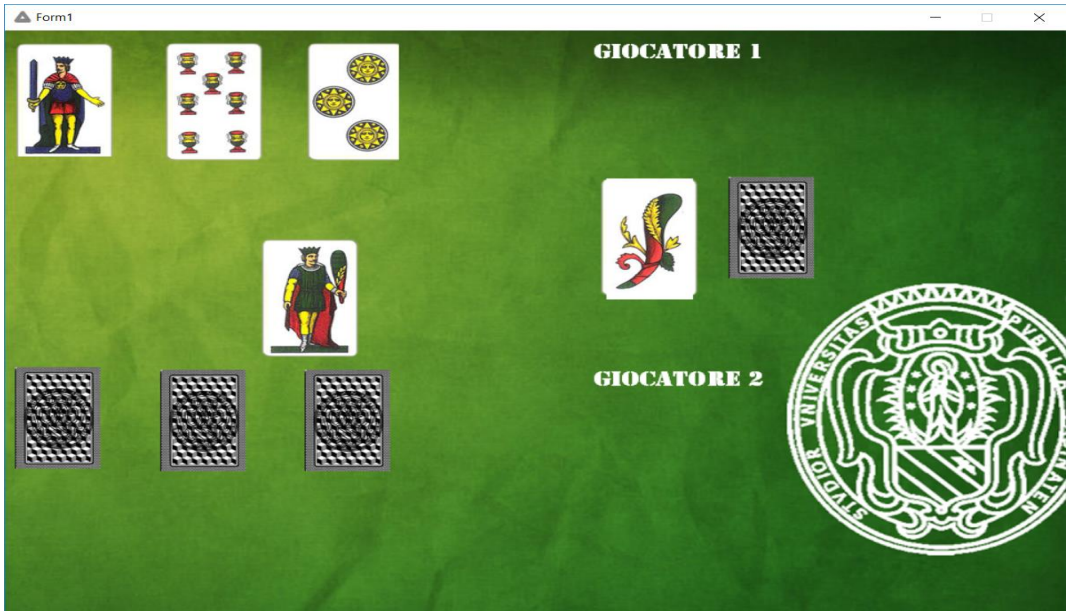
Si dà la possibilità, mediante dei button, di:

-Avviare il gioco.

-Consultare le istruzioni.

-Uscire dal gioco.

2)Il secondo Form introdotto è View; rappresenta il tavolo da gioco condiviso tra i due giocatori. Le tre pictureBox sopra rappresentano le carte del mazzo di giocatore1. Le tre in basso rappresentano le carte della mano di giocatore2.



A destra invece viene rappresentato sempre mediante pictureBox il mazzo mischiato e la carta rappresentante la briscola.
3)Il terzo Form introdotto è Fine che appare solo a fine partita e invita l'utente a rigiocare oppure ad uscire dal gioco con un interfaccia che varia a seconda se la partita è finita in pareggio o meno.

3.4 Design Pattern

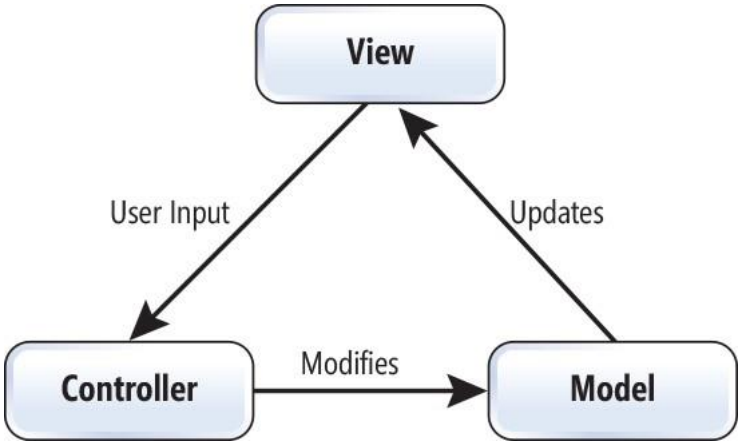
Per lo sviluppo di questo software si utilizza il pattern di sviluppo MVC(model view control).

L'utilizzo di questo tipo di pattern non è casuale, infatti permette di avere un distacco meglio di ogni altro, tra la parte logica e l'interfaccia.

Così facendo, un ipotetico aggiornamento dell'interfaccia sarà più rapido e meno rischioso perché non altera la parte di codice dove risiede la logica principale del gioco.

Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali: il model fornisce i metodi per accedere ai dati utili all'applicazione; il view visualizza i dati contenuti nel model e si occupa dell'interazione con utenti; il controller riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti.

In questo caso infatti l'utente visualizzerà le carte da poter scegliere e potrà decidere quale carta buttare tramite la classe control che a seconda delle azioni svolte in partita aggiorna in seguito le componenti del form, come punteggio o in generale le picturebox dopo ogni carta gettata o pescata.



3.5 Organizzazione progetto

Il progetto è organizzato con la combinazione di due sottoprogetti:

Il primo progetto chiamato GiocoCarte si occupa di descrivere la parte logica e di essere quindi il Model dell'applicazione con la creazione delle classi necessarie quali:

Carte con le classi ereditate CarteLisci e CartePunti;

Mazzo;

Tavolo;

Giocatore;

Partita;

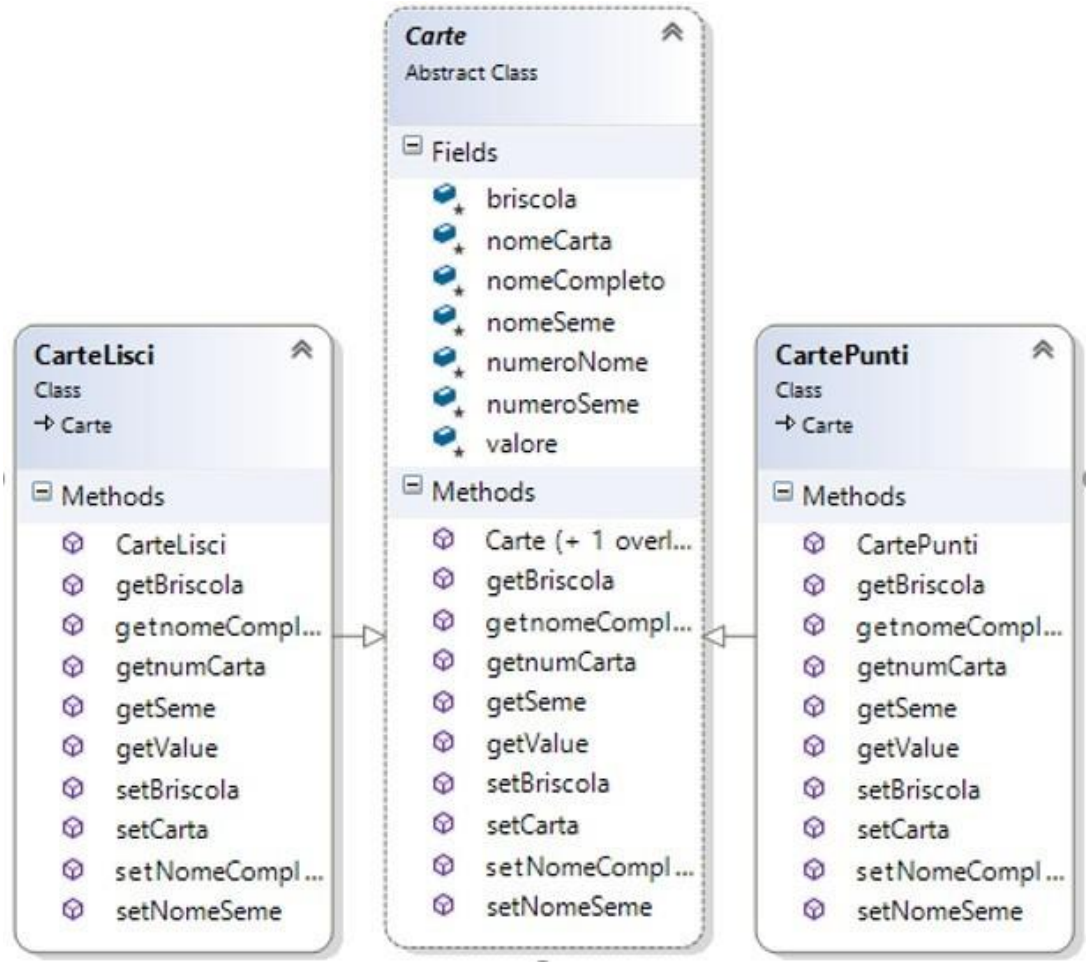
Il secondo progetto chiamato BriscolaX2 si occupa in generale di dare un interfaccia grafica a ciò che viene manipolato in GiocoCarte. L'insieme dei form visti nel capitoletto 3.3 rappresentano il View e invece la classe Control rappresenta il Control del pattern descritto in precedenza.

Vedremo poi da vicino ogni classe e ne spiegheremo il funzionamento

4 Classi

Si descrivono di seguito le classi istanziate e utilizzate per lo sviluppo del progetto attraverso l'utilizzo di diagrammi UML.

4.1 Carte



E' stato possibile utilizzare i concetti di ereditarietà e polimorfismo; la classe Carte eredita due classi figlie chiamate CarteLisci e CartePunti e nel primo caso si crea una carta di tipo liscio, ovvero quelle con valore uguale a 0 e nel secondo caso invece si crea una carta con valore diverso da zero dipendentemente dal nome della carta come per esempio "al 3 si associa il valore 10", "l'asso vale 11" e così via coerentemente alla precedente tabella.

E’ stato sfruttato il concetto di polimorfismo dove la classe base può definire e implementare metodi virtuali e le classi derivate possono eseguirne l’override, ossia ne forniscono una propria definizione e implementazione. Durante la fase di esecuzione, quando il codice client chiama il metodo, cerca il tipo dell’oggetto e richiama quell’override del metodo virtuale. Nel codice sorgente è possibile chiamare un metodo su una classe base, causando l’esecuzione di una versione di tale metodo definita in una classe derivata.

L’override delle classi derivate può essere vista per quanto riguarda la funzione setCarta(), il quale output dipende dalla classe derivata scelta.

Coerentemente al concetto di override è stato sfruttato l’upcasting.

Ad esempio durante l’esecuzione del metodo mescolaMazzo presente nella classe mazzo, le carte non vengono trattate separatamente se lisci o punti, ma vengono intesi come derivati della classe base e non si lavora sulla classe derivata, ma sulla classe base Carte.

Un’altra caratteristica utilizzata in questa classe è l’incapsulamento, che rende gli attributi della classe corrente e delle derivate protetti dall’ esterno. L’oggetto carta ha dunque i seguenti attributi:

- nomeCarta che rappresenta il nome della carta in una stringa. Ad esempio "due", "tre", "asso" ecc.
- nomeSeme che rappresenta il nome del seme della carta in una stringa. Come "Oro", "Bastoni", "Coppe", "Spade".
- nomeCompleto che è composto dal nomeCarta e dal nomeSeme.
- numeroNome e numeroSeme sono due interi che servono in seguito alla creazione delle stringhe nomeCarta e di nomeSeme.
- un altro intero rappresenta invece il valore che viene assegnato alla carta .
- un booleano chiamato briscola invece che se settato a true conferma che la carta è una briscola.

Ha anche due metodi costruttori:

- Carte(int numeroNome,int numeroSeme) che serve a creare un oggetto carta con un certo nome e un certo seme.
- Carte() crea invece un oggetto carta senza nessun attributo inizializzato.

Infine i metodi sono:

- setCarta() : void che setta la stringa nomeCarta e il valore della carta.
- setNomeSeme() : void che a seconda del valore di numeroSeme setta la stringa nomeSeme.
- setNomeCompleto(): void che setta il nomeCompleto.
- setBriscola(): void che imposta il booleano briscola a true.
- getSeme() : int che rilascia il valore di numeroSeme.
- getBriscola() : bool rilascia invece il valore di briscola.
- getValue(): int rilascia il valore della carta in un intero.
- getNumCarta() : int rilascia in un intero il numeroNome.
- getnomeCompleto() : string rilascia in una stringa il nomeCompleto.

```
using System;
using System.Collections.Generic;
using System.Linq; using System.Text;
using System.Threading.Tasks;
namespace LogicaCarte
{
    public abstract class Carte
    {
        protected String nomeCarta;
        protected String nomeCompleto;
        protected String nomeSeme;
        protected int numeroNome;
        protected int valore;
        protected int numeroSeme;
        protected Boolean briscola;
```

```
public Carte(int i, int j)
{
        this.numeroNome = j;
        this.numeroSeme = i;
        this.briscola = false;
}

public Carte()
{
}
public virtual void setCarta()
{
}
public virtual void setNomeSeme()
{
}
public virtual void setNomeCompleto()
{
}
public virtual int getSeme()
{
        return numeroSeme;
}
public virtual void setBriscola()
{
```

```
            this.briscola = true;
        }
        public virtual Boolean getBriscola()
        {
            return this.briscola;
        }
        public virtual int getValue()
        {
            return this.valore;
        }
        public virtual int getnumCarta()
        {
            return this.numeroNome;
        }
        public virtual String getnomeCompleto()
        {
            return this.nomeCompleto;
        }
    }
}
```

4.2 CarteLisci e CartePunti

La differenza sostanziale tra carteLisci e CartePunti si può notare nei seguenti frammenti di codice focalizzandoci sull'override delle funzioni setCarta() che rappresentano la spiegazione a riguardo del polimorfismo vista precedentemente.

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace LogicaCarte
{
    class CarteLisci : Carte
    {
        public CarteLisci(int i,int j) : base(i, j)
        {
        }
        public override void setCarta()
        {
            switch (numeroNome)
            {
                case 2:
                valore = 0; nomeCarta = "Due";
                    break;
                case 4:
                valore = 0; nomeCarta = "Quattro";
                    break;
                case 5:
                valore = 0; nomeCarta = "Cinque";
                    break;
                case 6:
                valore = 0; nomeCarta = "Sei";
                    break;
```

```
        case 7: valore = 0; nomeCarta = "Sette";
                break;
        }
}
public override void setNomeSeme()
{
        switch (numeroSeme)
        {
                case 1: nomeSeme = "Oro";
                        break;
                case 2: nomeSeme = "Coppe";
                        break;
                case 3: nomeSeme = "Bastoni";
                        break;
                case 4: nomeSeme = "Spade";
                        break;
        }
}
public override void setNomeCompleto()
{
        nomeCompleto = nomeCarta + nomeSeme;
}
public override int getSeme()
{
        return numeroSeme;
}
```

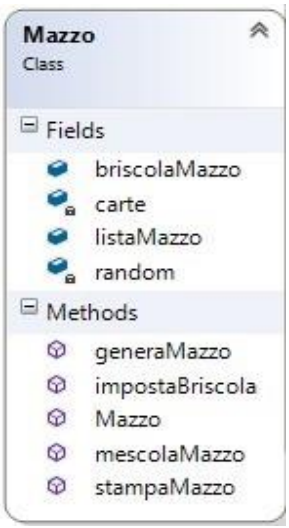
```
        public override void setBriscola()
        {
            briscola = true;
        }
        public override Boolean getBriscola()
        {
            return this.briscola;
        }
        public override int getValue()
        {
            return this.valore;
        }
        public override int getnumCarta()
        {
            return this.numeroNome;
        }
        public override String getnomeCompleto()
        {
            return this.nomeCompleto;
        }
    }
}
```

4.3 Mazzo

Il mazzo vero e proprio è inteso come una lista di oggetti di tipo Carte.

La classe mazzo descrive l'oggetto Mazzo e può generare un mazzo di carte grazie al metodo generaMazzo() e di mescolarlo grazie al metodo mescolaMazzo().

Quest'ultimo non fa altro che scorrere la lista di carte e le inverte con altre in posizioni random. Inoltre una funzione chiamata impostaBriscola imposta le briscole del mazzo.



Di seguito vi è l'implementazione:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace LogicaCarte
```

```
{
    public class Mazzo
    {
        Carte carte;
        public List<Carte> listaMazzo;
        Random random;
        public Carte briscolaMazzo;
        public Mazzo()
        {
            this.listaMazzo = new
            List<Carte>();
            this.random = new
            Random();
        }
        public void generaMazzo()
        {
            for(int i = 1; i <= 4; i++)
            {
                for(int j = 1; j <= 10;
                j++)
                {
                    //crea una
                    nuova istanza
                    di tipo carta
                    if(j == 2 || j == 4 || j == 5 || j == 6 || j == 7)
                    {
                        this.carte = new CarteLisci(i,j);
                        this.carte.setCarta();
```

```
                    this.carte.setNomeSeme();
                    this.carte.setNomeCompleto();
                }

                if (j == 1|| j== 3 || j == 8 || j == 9 || j == 10)
                {

                    this.carte = new CartePunti(i, j);
                    this.carte.setCarta();
                    this.carte.setNomeSeme();
                    this.carte.setNomeCompleto();
                }

                this.listaMazzo.Add(carte);
            }

        }

}

//funzione mescolaMazzo
public void mescolaMazzo()
{

        int posizCasuale;

        for (int i = this.listaMazzo.Count; i > 0; i--)
        {
            posizCasuale = random.Next(i - 1);
            Carte daCambiare;
        try
        {
            daCambiare = this.listaMazzo[i];
            this.listaMazzo[i] = this.listaMazzo[posizCasuale];
            this.listaMazzo[posizCasuale] = daCambiare;
        }
        catch (System.ArgumentOutOfRangeException)
        {

        }

}
```

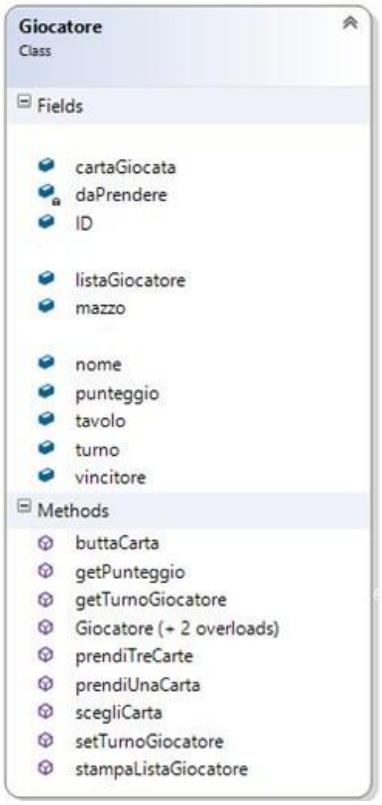
```
        //scambio di posto le carte..
        }
        public void impostaBriscola()
        {
                int a = 0;
                briscolaMazzo = this.listaMazzo.FirstOrDefault();
                this.listaMazzo.Remove(briscolaMazzo);
                this.listaMazzo.Add(briscolaMazzo);

                Console.WriteLine(" BRISCOLA "+briscolaMazzo.getnomeCompleto());

                for(int i = 0; i < this.listaMazzo.Count; i++)
                {
                        Carte corrente = this.listaMazzo[i];
                        if(corrente.getSeme() ==
                        briscolaMazzo.getSeme())
                {
                        this.listaMazzo[i].setBriscola();
                        Console.WriteLine(" " +listaMazzo[i].getnomeCompleto());
                        a++;
                }
        }
        Console.WriteLine("Esistono " + a + " briscole");
}
public void stampaMazzo()
{
        for(int i = 0; i < listaMazzo.Count; i++)
        {
                Console.WriteLine(" " + listaMazzo[i].getnomeCompleto()); }
        }
```

}

4.4 Giocatore



I seguenti attributi:

- ID che è un intero e rappresenta il giocatore.
- nome, che è una stringa associata al nome del giocatore.
- listaGiocatore invece è una lista di carte che rappresenta il mazzo di carte che ha in mano il giocatore.
- turno è un bool che se settato a true autorizza al giocatore a fare una mossa .
- mazzo invece è un oggetto di tipo Mazzo che rappresenta il mazzo di carte della partita.
- tavolo invece è un oggetto di tipo Tavolo.
- cartaGiocata rappresenta un oggetto di tipo carta, ovvero la carta scelta dal giocatore da essere buttata.

- daPrendere è un oggetto di tipo carta che serve da appoggio per rilasciare la carta pescata dal giocatore.
- punteggio è un intero che incrementa ogni presa da parte del giocatore.

I due metodi costruttori usati sono:

- Giocatore(int,Mazzo,Tavolo) che istanzia un oggetto con un determinato ID, mazzo e tavolo.

-Giocatore(string,Mazzo,Tavolo) che istanzia un oggetto con un determinato Nome, mazzo e tavolo.

I metodi sono invece:

- setTurnoGiocatore(bool) : void che setta il valore del booleano turno.
- getTurnoGiocatore() : bool che rilascia il valore della variabile turno.
- prendiUnaCarta() : Carte che rilascia la carta pescata.
- prendiTreCarte() : void, aggiunge le prime tre carte pescate dal mazzo durante il primo turno alla listaGiocatore.
- scegliCarta(Carte) : Carte setta e rilascia cartaGiocata.
- buttaCarta() : void che butta la carta selezionata, quindi cartaGiocata. - stampaListaGiocatore() : void che stampa le carte che il giocatore ha in mano.
- getPunteggio() : int rilascia il punteggio del giocatore.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace LogicaCarte
{
    public class Giocatore
    {
```

```
public int ID;
public String nome;
public List<Carte> listaGiocatore;
public Boolean turno;
public Mazzo mazzo;
public Tavolo tavolo;
public Carte cartaGiocata;
Carte daPrendere;
public int punteggio;
public Giocatore(int ID,Mazzo mazzo,Tavolo tavolo)
{
        this.ID = ID;
        this.turno = false;
        this.listaGiocatore = new List<Carte>();
        this.mazzo = mazzo;
        this.tavolo = tavolo;
        this.punteggio = 0;
        // this.a = 0;
}
public Giocatore(String nome,Mazzo mazzo,Tavolo tavolo)
{
        this.nome = nome;
        this.turno = false;
        this.listaGiocatore = new List<Carte>();
        this.mazzo = mazzo;
        this.tavolo = tavolo;
```

```
        this.punteggio = 0;
}
public void setTurnoGiocatore(Boolean turno)
{
        this.turno = turno;
}
public Boolean getTurnoGiocatore()
{
        return this.turno;
}
//funzione che seleziona una carta dal proprio mazzo
//funzione che prende tre carte dal mazzo principale(ALL'INIZIO)
public void prendiTreCarte()
{
        Carte daPrendere;
        for(int i = 0; i < 3; i++)
        {
                daPrendere = prendiUnaCarta();
                Console.WriteLine(" "+ID+" prende "
                + daPrendere.getnomeCompleto());
        }
}
//funzione che prende una carta dal mazzo principale(OGNI PRESA)
public Carte prendiUnaCarta()
{
        if (this.mazzo.listaMazzo.Count == 0)
```

```
        {
            //non posso prendere la prossima carta quindi
        }
        else
        {
            daPrendere = this.mazzo.listaMazzo.FirstOrDefault();
            this.mazzo.listaMazzo.Remove(daPrendere);
            this.listaGiocatore.Add(daPrendere);
        }
        return daPrendere;
    }
    //funzione che butta una carta dal proprio mazzo
    public Carte scegliCarta(Carte carte)
    {
        this.cartaGiocata = carte;
        return cartaGiocata;
    }
    public void buttaCarta()
    {
        //se numTurno È = 0...all'inizio
        if(tavolo.numTurno == 0)
        {
            tavolo.setPrimo(this);
            tavolo.numTurno++;
            this.listaGiocatore.Remove(cartaGiocata);
        }
        else if(tavolo.numTurno == 1)
```

```
                {
                        tavolo.setSecondo(this);
                        this.listaGiocatore.Remove(cartaGiocata);
                        //e si fa il confronto
                        this.tavolo.confrontaCarte(tavolo.primo, tavolo.secondo);
                }
        }
        public void stampaListaGiocatore()
        {
                for(int i = 0; i < this.listaGiocatore.Count; i++)
                {
                        Console.WriteLine("
                        "+listaGiocatore[i].getnomeCompleto());
                }
        }
        public int getPunteggio()
        {
                return punteggio;
        }
    }
}
```

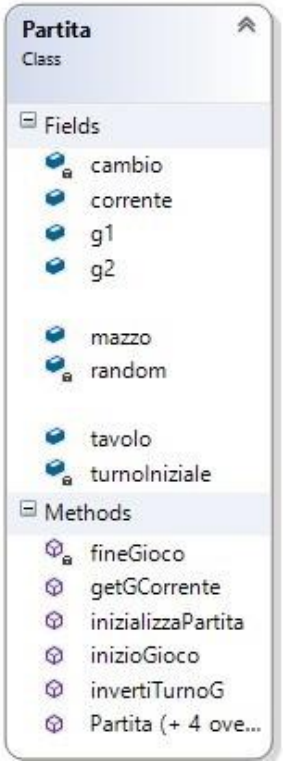
È presente il concetto di overload dei metodi, ovvero la possibiltà di definire più versioni dello stesso metodo.

Infatti, un oggetto giocatore può essere istanziato tramite due metodi costruttori diversi, che sono separati nel programma compilato.

Un metodo che riceve un parametro stringa ideato per permettere all'utente di creare un Giocatore avente come id un nome, separato da uno che riceve come id un parametro intero predefinito.

4.5 Partita

La classe partita gestisce la logica del gioco.



Sono presenti i seguenti attributi :

- Tavolo: oggetto di tipo Tavolo.
- random è un oggetto della classe Random che serve a settare il turno iniziale con probabilità 50 e 50.
- mazzo rappresenta il mazzo della partita.
- g1 è un oggetto di tipo Giocatore che rappresenta il Giocatore 1.
- g2 rappresenta invece il Giocatore 2.
- cambio è una variabile contatore intera che se assume valore 1 allora si invoca la funzione che inverte il turno del giocatore corrente.
- corrente è un oggetto di tipo Giocatore che tiene traccia del giocatore corrente della partita.
- turnoIniziale è una variabile intera che setta chi il giocatore corrente iniziale.

C’è solo un metodo costruttore :

- Partita().

Altri metodi sono:

- inizializzaPartita() : void Ł un metoo che inizializza la partita.
- - inizioGioco() :void e neGioco() : void sono delle funzioni che rappresentano l’intero gioco debuggato se lanciato da terminale.
- getGCorrente(Giocatore,Giocatore) : Giocatore prende in input due Giocatori e rilascia quello con turno settato a true.
- invertiTurnoG(Giocatore,Giocatore) : void prende in input due Giocatori e scambia i valori della variabile turno. Ci serve a cambiare turno.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace LogicaCarte
{
        public class Partita
        {
                public Tavolo tavolo;
                Random random;
                public Mazzo mazzo;
                public Giocatore g1;
                public Giocatore g2;
                public int cambio;
                public Giocatore corrente;
                int turnoIniziale;
```

```
public Partita()
{
        tavolo = new Tavolo();
        mazzo = new Mazzo();
        g1 = new Giocatore(1, mazzo, tavolo);
        g2 = new Giocatore(2, mazzo, tavolo);
        random = new Random();
        turnoIniziale = 0;
        cambio = 0;
}
public void inizializzaPartita()
{
        //per prima cosa setto il turno di chi è di mano
        //in maniera random
        turnoIniziale = random.Next(2);
        if (turnoIniziale == 0)
        {
                g1.setTurnoGiocatore(true);
                g2.setTurnoGiocatore(false);
        } else if (turnoIniziale == 1)
        {
                g1.setTurnoGiocatore(false);
                g2.setTurnoGiocatore(true);
        }
        mazzo.generaMazzo();
        mazzo.mescolaMazzo();
        mazzo.stampaMazzo();
        mazzo.impostaBriscola();
        if (turnoIniziale == 0)
        {
                g1.prendiTreCarte();
                g2.prendiTreCarte();
        } else if (turnoIniziale == 1)
        {
                g2.prendiTreCarte();
```

```
                g1.prendiTreCarte();
        }
}
//la seguente funzione viene
//lanciata solo nell'applicazione
//da terminale essendo una vera e
//propria forma di debug
public void inizioGioco()
{
        Carte scelta;
        while(mazzo.listaMazzo.Count > 0)
        {
                corrente = getGCorrente(g1, g2);
                Console.WriteLine("Tocca a " + corrente.ID);
                //sceglie la carta tra quelle che ha in lista
                Console.WriteLine("inserire 0, 1 o 2 per scegliere "
                +" una delle carte della lista");
                int index = int.Parse(Console.ReadLine());
                scelta = corrente.scegliCarta(corrente.listaGiocatore[index]);
                Console.WriteLine("hai scelto "
                + corrente.cartaGiocata.getnomeCompleto());
                Console.WriteLine("hai buttato "
                + corrente.cartaGiocata.getnomeCompleto());
                corrente.buttaCarta(); cambio++;
                if (cambio == 1)
                {
                        invertiTurnoG(g1, g2);
                }
                corrente = getGCorrente(g1, g2);
                Console.WriteLine("Tocca a " + corrente.ID);
                //sceglie la carta tra quelle che ha in lista
                Console.WriteLine("inserire 0, 1 o 2 per scegliere "
                +" una delle carte della lista");
                index = int.Parse(Console.ReadLine());
                scelta = corrente.scegliCarta(corrente.listaGiocatore[index]);
                Console.WriteLine("hai scelto "
                + corrente.cartaGiocata.getnomeCompleto());
```

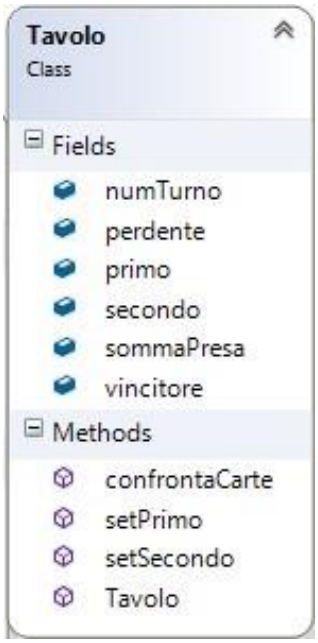
```
                Console.WriteLine("hai buttato "
                + corrente.cartaGiocata.getnomeCompleto());
                corrente.buttaCarta();
                //aggiorno il turno e prendo una carta corrente =
                getGCorrente(g1, g2);
                Console.WriteLine("Pu    prendere una carta " + corrente.ID);
                Carte presaCorr = corrente.prendiUnaCarta();
                Console.WriteLine("" + presaCorr.getnomeCompleto());
                Console.WriteLine("Stampa lista giocatore "+corrente.ID
                +" corrente :");
                corrente.stampaListaGiocatore(); invertiTurnoG(g1, g2);
                corrente = getGCorrente(g1,g2); presaCorr =
                corrente.prendiUnaCarta();
                Console.WriteLine("" + presaCorr.getnomeCompleto());
                Console.WriteLine("Stampa lista giocatore " + corrente.ID
                + " corrente :");
                corrente.stampaListaGiocatore(); invertiTurnoG(g1, g2);
                cambio = 0;
                }
fineGioco();
}
private void fineGioco()
{
        if(g1.getPunteggio() > g2.getPunteggio())
        {
                Console.WriteLine("Vince " + g1.ID + " per " +
                g1.getPunteggio()
                + "a " + g2.getPunteggio());
        }else if (g1.getPunteggio() < g2.getPunteggio())
        {
                Console.WriteLine("Vince " + g2.ID + " per " +
                g2.getPunteggio()
                + "a " + g1.getPunteggio());
        }else if(g1.getPunteggio() == g2.getPunteggio())
        {
                Console.WriteLine("Pareggio " + g1.getPunteggio()
                + " a " + g2.getPunteggio());
```

```
                }
            }


            //funzione che ritorna il giocatore che può effettuare la mossa
            public Giocatore getGCorrente(Giocatore g1,Giocatore g2)
            {
                if (g1.getTurnoGiocatore())
                {
                    corrente = g1;
                }else if (g2.getTurnoGiocatore())
                {
                    corrente = g2;
                }
                return corrente;
            }
            public void invertiTurnoG(Giocatore g1,Giocatore g2)
            {
                if(corrente == g1)
                {
                    g1.setTurnoGiocatore(false);
                    g2.setTurnoGiocatore(true);
                }else if(corrente == g2)
                {
                    g2.setTurnoGiocatore(false);
                    g1.setTurnoGiocatore(true);
                }
            }
        }
}
```

4.6 Tavolo

La classe Tavolo è una classe utilizzata per di svolgere determinati compiti quali il confronto delle carte buttate dai giocatori sul tavolo e l'associazione quindi di un vincitore e di un perdente alla mano appena giocata.



Come si evince dallo schema UML sono stati utilizzati i seguenti attributi: vincitore che è un oggetto della classe Giocatore che verrà settato alla fine del confronto delle carte con il giocatore che ha vinto la mano.

- perdente è sempre un oggetto della classe Giocatore che verrà invece settato con il giocatore che ha perso la mano corrente.
- primo e secondo sono due attributi sempre di tipo Giocatore che invece tengono traccia dell'ordine in cui i giocatori hanno tirato le carte al fine di permettere il confronto.
- numturno è un oggetto di tipo intero che invece determina chi è il primo a tirare.
- sommaPresa invece è una variabile sempre di tipo intero che rappresenta la somma dei valori delle carte giocate al ne poi di sommarla al punteggio del giocatore vincente.

C'è solo un metodo costruttore che inizializza i parametri ed Ł - Tavolo().
I metodi invece di cui usufruiamo sono:

- setPrimo(Giocatore) : void che setta il giocatore come primo.
- setSecondo(Giocatore) : void che setta il giocatore come secondo.
- ConfrontaCarte(Giocatore, Giocatore) : void che confronta le carte sul tavolo tramite dei controlli if e else if.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace LogicaCarte
{
	public class Tavolo
	{
		public Giocatore vincitore;
		public Giocatore perdente;
		public Giocatore primo;
		public Giocatore secondo;
		public int numTurno;
		public int sommaPresa;
		public Tavolo()
		{
			this.numTurno = 0;
			this.sommaPresa = 0;
		}
		//ritorna il giocatore che vince
		public void setPrimo(Giocatore p)
		{
			primo = p;
		}
		public void setSecondo(Giocatore p)
		{
			secondo = p;
		}
		public void confrontaCarte(Giocatore primo, Giocatore secondo)
		{
```

```
        Console.WriteLine(" " + primo.cartaGiocata.getnomeCompleto()
        + " " + primo.cartaGiocata.getValue());
        Console.WriteLine(" " + secondo.cartaGiocata.getnomeCompleto()
        + " " + secondo.cartaGiocata.getValue());
        sommaPresa = primo.cartaGiocata.getValue()
        + secondo.cartaGiocata.getValue();
        //SEMI DIVERSI
        if (primo.cartaGiocata.getSeme() !=
        secondo.cartaGiocata.getSeme())
        {
                // MessageBox.Show("SEMI DIVERSI");
                //CASO CHE SOLO LA PRIMA E' BRISCOLA
                if ((primo.cartaGiocata.getBriscola()) &&
                (secondo.cartaGiocata.getBriscola() == false))
                {
                //MessageBox.Show("PRIMA BRISCOLA SECONDA NO");
                //vince la prima carta
                Console.WriteLine("Vince " + primo.ID);
                primo.punteggio += sommaPresa;
                vincitore = primo;
                perdente = secondo;
        }
//CASO CHE SOLO LA SECONDA E'BRISCOLA
else if ((primo.cartaGiocata.getBriscola() == false) &&
(secondo.cartaGiocata.getBriscola()))
{
        //vince la seconda carta
        //MessageBox.Show("PRIMA no SECONDA Briscola");
        Console.WriteLine("Vince " + secondo.ID);
        secondo.punteggio += sommaPresa;
        vincitore = secondo;
        perdente = primo;
}
//CASO CHE ENTRAMBE NON SONO BRISCOLE
else if ((primo.cartaGiocata.getBriscola() == false) &&
(secondo.cartaGiocata.getBriscola() == false))
{
```

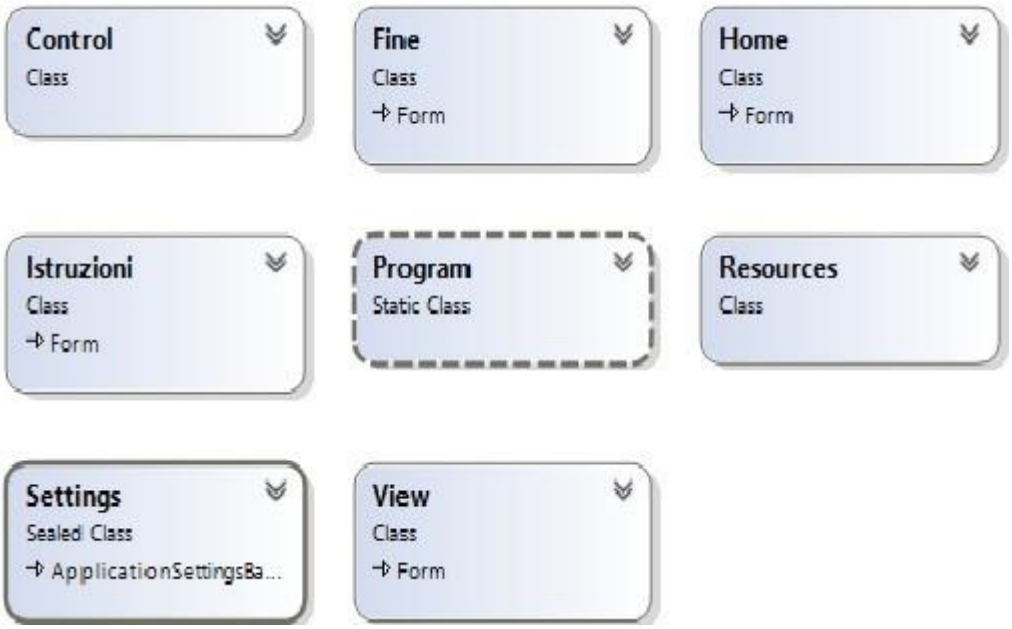
```
                //vince la prima carta
                //MessageBox.Show("PRIMA no SECONDA NO");
                Console.WriteLine("Vince " + primo.ID);
                primo.punteggio += sommaPresa;
                vincitore = primo;
                perdente = secondo;
        }
}
//SEMI UGUALI
else if (primo.cartaGiocata.getSeme() == secondo.cartaGiocata.getSeme())
{
        //MessageBox.Show("SEMI UGUALI");
        //CASO CHE ENTRAMBE SONO BRISCOLE-> vince il valore più grande
        if (primo.cartaGiocata.getBriscola() &&
        secondo.cartaGiocata.getBriscola())
        {
                //MessageBox.Show("ENTRAMBE BRISCOLE");
                if (primo.cartaGiocata.getValue() >
                secondo.cartaGiocata.getValue())
                {
                        //vince il primo
                        Console.WriteLine("Vince " + primo.ID);
                        primo.punteggio += sommaPresa;
                        vincitore = primo;
                        perdente = secondo;
                }
                else if (primo.cartaGiocata.getValue() <
                secondo.cartaGiocata.getValue())
                {
                        //vince il secondo
                        Console.WriteLine("Vince " + secondo.ID);
                        secondo.punteggio += sommaPresa;
                        vincitore = secondo;
                        perdente = primo;
                }
                else if (primo.cartaGiocata.getValue() ==
                secondo.cartaGiocata.getValue())
```

```
        {
        //MessageBox.Show("PRIMA = SECONDA");
        //nel caso siano uguali i valori e i semi uguali....
        //vuol dire che ci troviamo in presenza di due lisci
        //tra due lisci vince quello per distinguerlo con numCart
        if (primo.cartaGiocata.getnumCarta() >
        secondo.cartaGiocata.getnumCarta())
        {
                //vince il primo
                Console.WriteLine("Vince " + primo.ID);
                primo.punteggio += sommaPresa;
                vincitore = primo;
                perdente = secondo;
        }
else if (primo.cartaGiocata.getnumCarta() <
secondo.cartaGiocata.getnumCarta())
{
        //vince il secondo
        Console.WriteLine("Vince " + secondo.ID);
        secondo.punteggio += sommaPresa;
        vincitore = secondo;
        perdente = primo;
}
else if (primo.cartaGiocata.getBriscola() == false &&
secondo.cartaGiocata.getBriscola() == false)
{
        // MessageBox.Show("PRIMO no,SECONDA no");
        if (primo.cartaGiocata.getValue() >
        secondo.cartaGiocata.getValue())
        {
                //vince il primo
                Console.WriteLine("Vince " + primo.ID);
                primo.punteggio += sommaPresa;
                vincitore = primo;
                perdente = secondo;
        }
```

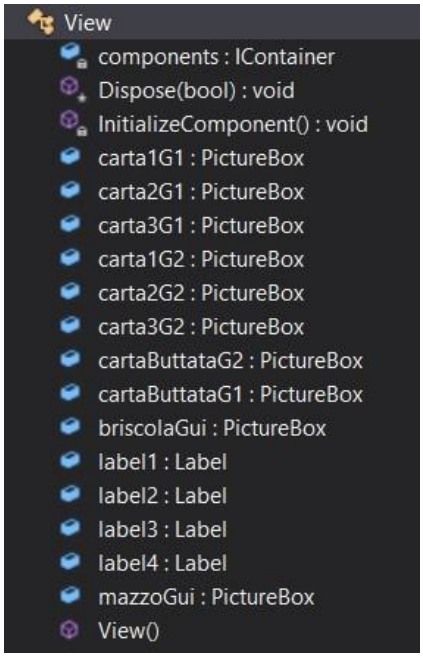
```
                else if (primo.cartaGiocata.getValue() <
                secondo.cartaGiocata.getValue())
                {
                        //vince il secondo
                        Console.WriteLine("Vince " + secondo.ID);
                        secondo.punteggio += sommaPresa;
                        vincitore = secondo;
                        perdente = primo;
                }
                else if (primo.cartaGiocata.getValue() ==
                secondo.cartaGiocata.getValue())
                {
                        //nel caso siano uguali i valori e i semi uguali....
                        //vuol dire che ci troviamo in presenza di due lisci
                        //tra due lisci vince quello per distinguerlo con numCart
                        if (primo.cartaGiocata.getnumCarta() >
                        secondo.cartaGiocata.getnumCarta())
                        {
                                //vince il primo
                                Console.WriteLine("Vince " + primo.ID);
                                primo.punteggio += sommaPresa;
                                vincitore = primo;
                                perdente = secondo;
                        }
                        if (primo.cartaGiocata.getnumCarta() <
                        secondo.cartaGiocata.getnumCarta())
                        {
                                //vince il secondo
                                Console.WriteLine("Vince " + secondo.ID);
                                secondo.punteggio += sommaPresa;
                                vincitore = secondo;
                                perdente = primo;
                        }
                }
        }
}
//funzione che ogni volta che un giocatore tira la carta
```

```
        //aumenta un contatore numTurno.
        //appena numTurno == 2 allora vorrà dire che
        this.numTurno = 0;
        vincitore.setTurnoGiocatore(true);
        perdente.setTurnoGiocatore(false);
        }
}
```

Adesso vediamo le classi che compongono invece la parte grafica dell'applicazione, mostrandole prima nel seguente schema:



4.7 View



Rappresenta il tavolo da gioco composto da 10 pictureBox:

- 6 sono utilizzate per la rappresentazione dei due mazzi dei giocatori, quindi una per ogni carta.
- 2 rappresentano invece le carte giocate dai giocatori.
- 1 rappresenta la briscola che viene girata all'inizio della partita e che sarà l' ultima carta ad essere presa.
- 1 rappresenta il mazzo vero e proprio.

e 4 label:

- 2 usati per mostrare all'utente una dove si trova giocatore 1 e l'altra per giocatore 2.(label1 e label2)
- 2 invece usati per tenere traccia dei punteggi dei giocatori.(label3 e label4)

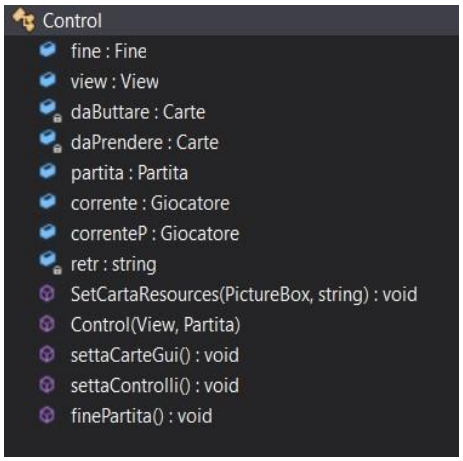
E' presente un solo metodo costruttore :

- View() che inizializza il form.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace briscolax2
{
    public partial class View : Form
    {
        public View()
        {
            InitializeComponent();
            this.label3.Text = null;
            this.label4.Text = null;
        }
    }
}
```

4.8 Control

Control è la classe che rappresenta il controller della partita infatti ad ogni cambiamento presente in partita permette di aggiornare il form visto dall'utente.



Gli attributi presenti sono i seguenti:

- fine è un oggetto di tipo Fine dove un form che compare alla fine della partita.
- view che è di tipo View e che rappresenta quindi la grafica del tavolo da gioco.
- daButtare e daPrendere che sono due oggetti di tipo Carte che servono da appoggio per le funzioni "scegliCarta" e "prendiUnaCarta".
- corrente è un oggetto di tipo giocatore che tiene traccia del giocatore corrente e correnteP invece tiene traccia del giocatore che ha perso la mano precedente.

L'unico metodo costruttore è

- Control(View,Partita) che prende quindi in input il form da aggiornare e la partita.

I metodi usati sono invece:

- setCartaResources(PictureBox image,string nomeCarta) : una funzione che setta la pictureBox presa in input con l'immagine che risiede nelle resources associata alla stringa nomeCarta.
- settaCarteGui() : void che controlla il comportamento della partita e aggiorna le pictureBox delle carte giocatori.
- settaControlli() : void che invece setta le azioni da compiere una volta fatto click nelle varie pictureBox che rappresentano le carte da poter giocare.
- nePartita() : void invece è una funzione che chiamata a ne partita dice chi è il vincitore e il perdente e la somma dei punti totalizzati dai due giocatori.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Drawing;
namespace briscolax2
{
	public class Control
	{
		public Fine fine;
		public View view;
		LogicaCarte.CartedaButtare;
		LogicaCarte.Carte daPrendere;
		public LogicaCarte.Partita partita;
		public LogicaCarte.Giocatore corrente;
		public LogicaCarte.Giocatore correnteP;
		String retr = "retr";
		public void SetCartaResources(PictureBox image,string nomeCarta)
		{
			switch (nomeCarta)
			{
				case "AssoOro": image.Image =
				briscolax2.Properties.Resources.AssoOro;
				break;
				case "AssoCoppe": image.Image =
				briscolax2.Properties.Resources.AssoCoppe;
				break;
				case "AssoBastoni":
					image.Image =
					briscolax2.Properties.Resources.AssoBastoni;
				break;
				case "AssoSpade":
					image.Image =
					briscolax2.Properties.Resources.AssoSpade;
				break;
```

```
case "DueOro":
                image.Image =
                briscolax2.Properties.Resources.DueOro;
        break;
case "DueCoppe":
                image.Image =
                briscolax2.Properties.Resources.DueCoppe;
        break;
case "DueBastoni":
                image.Image =
                briscolax2.Properties.Resources.DueBastoni;
        break;
case "DueSpade":
                image.Image =
                briscolax2.Properties.Resources.DueSpade;
        break;
case "TreOro":
                image.Image =
                briscolax2.Properties.Resources.TreOro;
        break;
case "TreCoppe":
                image.Image =
                briscolax2.Properties.Resources.TreCoppe;
        break;
case "TreBastoni":
                image.Image =
                briscolax2.Properties.Resources.TreBastoni;
        break;
case "TreSpade":
                image.Image =
                briscolax2.Properties.Resources.TreSpade;
        break;
case "QuattroOro":
                image.Image =
                briscolax2.Properties.Resources.QuattroOro;
        break;
case "QuattroCoppe":
```

```
                image.Image =
                briscolax2.Properties.Resources.QuattroCoppe;
            break;
        case "QuattroBastoni":
                image.Image =
                briscolax2.Properties.Resources.QuattroBastoni
                ;
            break;
        case "QuattroSpade":
                image.Image =
                briscolax2.Properties.Resources.QuattroSpade;
            break;
        case "CinqueOro":
                image.Image =
                briscolax2.Properties.Resources.CinqueOro;
            break;
        case "CinqueCoppe":
                image.Image =
                briscolax2.Properties.Resources.CinqueCoppe;
            break;
        case "CinqueBastoni":
                image.Image =
                briscolax2.Properties.Resources.CinqueBastoni;
            break;
        case "CinqueSpade":
                image.Image =
                briscolax2.Properties.Resources.CinqueSpade;
            break;

        case "SeiOro":
                image.Image =
                briscolax2.Properties.Resources.SeiOro;
            break;
        case "SeiCoppe":
                image.Image =
                briscolax2.Properties.Resources.SeiCoppe;
            break;
```

```
case "SeiBastoni":
                image.Image =
                briscolax2.Properties.Resources.SeiBastoni;
        break;
case "SeiSpade":
                image.Image =
                briscolax2.Properties.Resources.SeiSpade;
        break;
case "SetteOro":
                image.Image =
                briscolax2.Properties.Resources.SetteOro;
        break;
case "SetteCoppe":
                image.Image =
                briscolax2.Properties.Resources.SetteCoppe;
        break;
case "SetteBastoni":
                image.Image =
                briscolax2.Properties.Resources.SetteBastoni;
        break;
case "SetteSpade":
                image.Image =
                briscolax2.Properties.Resources.SetteSpade;
        break;
case "DonnaOro":
                image.Image =
                briscolax2.Properties.Resources.DonnaOro;
        break;


case "DonnaCoppe":
                image.Image =
                briscolax2.Properties.Resources.DonnaCoppe;
        break;
case "DonnaBastoni":
                image.Image =
                briscolax2.Properties.Resources.DonnaBastoni;
```

```
            break;
        case "DonnaSpade":
                    image.Image =
                    briscolax2.Properties.Resources.DonnaSpade;
            break;
        case "CavalloOro":
                    image.Image =
                    briscolax2.Properties.Resources.CavalloOro;
            break;
        case "CavalloCoppe":
                    image.Image =
                    briscolax2.Properties.Resources.CavalloCoppe;
            break;
        case "CavalloBastoni":
                    image.Image =
                    briscolax2.Properties.Resources.CavalloBastoni;
            break;
        case "CavalloSpade":
                    image.Image =
                    briscolax2.Properties.Resources.CavalloSpade;
            break;
        case "ReOro":
                    image.Image =
                    briscolax2.Properties.Resources.ReOro;
            break;
        case "ReCoppe":
                    image.Image =
                    briscolax2.Properties.Resources.ReCoppe;
            break;
        case "ReBastoni":
                    image.Image =
                    briscolax2.Properties.Resources.ReBastoni;
            break;
        case "ReSpade":
                    image.Image =
                    briscolax2.Properties.Resources.ReSpade;
            break;
```

```
                case "retr":
                        image.Image =
                        briscolax2.Properties.Resources.retr;
                    break;
                case "coppa":
                        image.Image =
                        briscolax2.Properties.Resources.coppa;
                    break;
                case "amicizia":
                        image.Image =
                        briscolax2.Properties.Resources.amicizia;
                        break;
                }
}
public Control(View view, LogicaCarte.Partita partita)
{
        this.view = view;
        this.partita = partita;
        SetCartaResources(view.briscolaGui,partita.mazzo.briscolaMazzo.
        getnomeCompleto());
        view.label1.Text = "Giocatore 1"; view.label2.Text = "Giocatore 2";
        SetCartaResources(view.mazzoGui, retr);
}
public void settaCarteGui()
{
        //controllo se il giocatore Ł il giocatore1
        corrente = partita.getGCorrente(partita.g1, partita.g2);
        //se giocatore1 è il giocatore corrente if (corrente == partita.g1)
        {
                //controllo quante carte ha in mano il giocatore,
                precisamente
                //se ne hameno di 3
                if (partita.g1.listaGiocatore.Count < 3)
                {
                        //controllo se il giocatore ha 2 carte in mano
                        if (partita.g1.listaGiocatore.Count == 2)
                        {
```

```
                    //setto le immagini da visualizzare
                    SetCartaResources(view.carta1G1, partita.g1.
                    listaGiocatore[0].getnomeCompleto());
                    SetCartaResources(view.carta2G1, partita.g1.
                    listaGiocatore[1].getnomeCompleto());
                    view.carta3G1.Image = null;
                }
            //controllo se il giocatore ha 1 carta in mano
            else if (partita.g1.listaGiocatore.Count == 1)
            {
                //setto le immagini da visualizzare
                SetCartaResources(view.carta1G1, partita.g1.
                listaGiocatore[0].getnomeCompleto());
                view.carta2G1.Image = null;
                view.carta3G1.Image = null;
            }
            //controllo se il giocatore non ha carte in mano
            else if (partita.g1.listaGiocatore.Count == 0)
            {
                //setto le immagini da visualizzare
                view.carta1G1.Image = null;
                view.carta2G1.Image = null;
                view.carta3G1.Image = null;
            }
        }
        //controllo se il giocatore ha 3 carte in mano
        else {
                //setto le immagini da visualizzare
                SetCartaResources(view.carta1G1, partita.g1.
                listaGiocatore[0].getnomeCompleto());
                SetCartaResources(view.carta2G1, partita.g1.
                listaGiocatore[1].getnomeCompleto());
                SetCartaResources(view.carta3G1, partita.g1.
                listaGiocatore[2].getnomeCompleto());
            }
        }
        //controllo se il giocatore è diverso dal corrente
```

```
else if (corrente != partita.g1)
{
        //controllo se il giocatore 1 ha meno di 3 carte
        if (partita.g1.listaGiocatore.Count < 3)
        {
                //se ha 2 carte
                if (partita.g1.listaGiocatore.Count == 2)
                {
                        //setto le immagini da visualizzare
                        view.carta3G1.Image = null;
                        SetCartaResources(view.carta1G1, retr);
                        SetCartaResources(view.carta2G1, retr);
                }
                //se ha 1 carta
                if (partita.g1.listaGiocatore.Count == 1)
                {
                        //setto le immagini da visualizzare
                        SetCartaResources(view.carta1G1, retr);
                        view.carta2G1.Image = null;
                        view.carta3G1.Image = null;
                }
                //se non ha carte
                if (partita.g1.listaGiocatore.Count == 0)
                {
                        //setto le immagini da visualizzare
                        view.carta1G1.Image = null;
                        view.carta2G1.Image = null;
                        view.carta3G1.Image = null;
                }
        }
//se ha 3 carte else
{
        //setto le immagini da visualizzare
        SetCartaResources(view.carta1G1, retr);
        SetCartaResources(view.carta2G1, retr);
        SetCartaResources(view.carta3G1, retr);
}
```

```
}
// se il giocatore Ł giocatore2
//se è anche il giocatore corrente
if (corrente == partita.g2)
{
        //se ha meno di 3 carte
        if (partita.g2.listaGiocatore.Count < 3)
        {
                //se ha 2 carte
                if (partita.g2.listaGiocatore.Count == 2)
                {
                        //setto le immagini da visualizzare
                        SetCartaResources(view.carta1G2, partita.g2.
                        listaGiocatore[0].getnomeCompleto());
                        SetCartaResources(view.carta2G2, partita.g2.
                        listaGiocatore[1].getnomeCompleto());
                        view.carta3G2.Image = null;
                }
                //se ha 1 carta
                else if (partita.g2.listaGiocatore.Count == 1)
                {
                        //setto le immagini da vsualizzare
                        SetCartaResources(view.carta1G2, partita.g2.
                        listaGiocatore[0].getnomeCompleto());
                        view.carta2G2.Image = null;
                        view.carta3G2.Image = null;
                }
                //se non ha carte
                else if (partita.g2.listaGiocatore.Count == 0)
                {
                        //setto le immagini da visualizzare
                        view.carta1G2.Image = null;
                        view.carta2G2.Image = null;
                        view.carta3G2.Image = null;
                }
        }
        //se ha invece 3 carte
```

```
        else
        {
            //setto le immagini da visualizzare
            SetCartaResources(view.carta1G2, partita.g2.
            listaGiocatore[0].getnomeCompleto());
            SetCartaResources(view.carta2G2, partita.g2.
            listaGiocatore[1].getnomeCompleto());
            SetCartaResources(view.carta3G2, partita.g2.
            listaGiocatore[2].getnomeCompleto());
        }
        //se il giocatore non è quello corrente
}
else if (corrente != partita.g2)
{
        //se ha meno di 3 carte
        if (partita.g1.listaGiocatore.Count < 3)
        {
            //se ha 2 carte in mano
            if (partita.g1.listaGiocatore.Count == 2)
            {
                //setto le immagini da visualizzare
                SetCartaResources(view.carta1G2, retr);
                SetCartaResources(view.carta2G2, retr);
                view.carta3G2.Image = null;
            }
            //se ho 1 carta
            if (partita.g1.listaGiocatore.Count == 1)
            {
                //setto le immagini da visualizzare
                SetCartaResources(view.carta1G2, retr);
                view.carta2G2.Image = null;
                view.carta3G2.Image = null;
            }
        //se non ho carte
        if (partita.g1.listaGiocatore.Count == 0)
        {
            //setto le immagini da visualizzare
```

```
                        view.carta1G2.Image = null;
                        view.carta2G2.Image = null;
                        view.carta3G2.Image = null;
                    }
                }
                //se ha 3 carte
                else
                {
                    //setto le immagini da visualizzare
                    SetCartaResources(view.carta1G2, retr);
                    SetCartaResources(view.carta2G2, retr);
                    SetCartaResources(view.carta3G2, retr);
                }
            }
        }
        public void settaControlli()
        {
            view.carta3G2.MouseClick += (s, es) =>
            {
                if (corrente == partita.g2)
                {
                    //scelgo la carta rappresentata dalla picturebox
                    daButtare = corrente.scegliCarta(corrente.listaGiocatore[2]);
                    //aggiorno il view
                    view.cartaButtataG2.Image = view.carta3G2.Image;
                    view.carta3G2.Image = null;
                    //se ha già tirato il primo giocatore
                    if (partita.tavolo.numTurno == 1)
                    {
                        //butto la carta selezionata e partirà il confronto
                        corrente.buttaCarta();
                        // setto adesso il giocatore corrente corrente =
                        partita.getGCorrente(partita.tavolo.vincitore,
                        partita.tavolo.perdente);
                        //se il mazzo non ha piø carte
                        if (partita.mazzo.listaMazzo.Count == 0)
                        {
```

```
            //aggiorno view
            view.carta1G2.Image = null;
            view.cartaButtataG1.Image = null;
            view.cartaButtataG2.Image = null;
            view.briscolaGui.Image = null;
            MessageBox.Show("Cambio turno a giocatore "
            + corrente.ID);
            //Addormento il thread per 3 secondi dando
            // quindi il tempo all'avversario per
            //posizionarsi davanti lo schermo senza guardare
            // le carte dell'altro giocatore
            //aggiorno view
            settaCarteGui(); finePartita();
        }
        //se invece il mazzo contiene ancora carte
        else
        {
            //prende una carta dal mazzo il giocatore corrente
            daPrendere = corrente.prendiUnaCarta();
            //prende una carta dal mazzo il giocatore
            //che ha perso nel turno precedente
            correnteP = partita.tavolo.perdente;
            daPrendere = correnteP.prendiUnaCarta();
            //aggiorno view view.cartaButtataG2.Image = null;
            view.cartaButtataG1.Image = null;
        }
        //aggiorno view passando ai label i punteggi
        //dei giocatori
        view.label3.Text = "" + partita.g1.getPunteggio();
        view.label4.Text = "" + partita.g2.getPunteggio();
        //setto giocatore corrente
        corrente = partita.getGCorrente(corrente, correnteP);
        //setto le immagini da visualizzare aggiornando il view
        SetCartaResources(view.carta1G2, retr);
        SetCartaResources(view.carta2G2, retr);
        SetCartaResources(view.carta3G2, retr);
```

```
                MessageBox.Show("Cambio turno a giocatore " +
                corrente.ID);
                settaCarteGui();
            }
            //se ancora si sta tirando per primo
            else if (partita.tavolo.numTurno == 0)
            {
                //setto il giocatore corrente della partita
                corrente = partita.getGCorrente(partita.g1, partita.g2);
                //butto la carta selezionata
                corrente.buttaCarta();
                //inverto il turno del giocatore
                partita.invertiTurnoG(partita.g1, partita.g2);
                //setto nuovamente il giocatore corrente della partita
                corrente = partita.getGCorrente(partita.g1, partita.g2);
                //visualizza e aggiorna view
                SetCartaResources(view.carta1G2, retr);
                SetCartaResources(view.carta2G2, retr);
                SetCartaResources(view.carta3G2, retr);
                MessageBox.Show("Cambio turno a giocatore " +
                corrente.ID);
                settaCarteGui();
            }
        }
};
view.carta2G2.MouseClick += (s, es) =>
{
        if (corrente == partita.g2)
        {
        //scelgo la carta rappresentata dalla picturebox
        daButtare = corrente.scegliCarta(corrente.listaGiocatore[1]);
        //aggiorno il view
        view.cartaButtataG2.Image = view.carta2G2.Image;
        view.carta2G2.Image = null;
        //se ha già tirato il primo giocatore
        if (partita.tavolo.numTurno == 1)
        {
```

```
//butto la carta selezionata e partirà il confronto
corrente.buttaCarta();
// setto adesso il giocatore corrente
corrente = partita.getGCorrente(partita.tavolo.vincitore,
partita.tavolo.perdente);
//se il mazzo non ha più carte
if (partita.mazzo.listaMazzo.Count == 0)
{
        //aggiorno view view.carta2G2.Image = null;
        view.cartaButtataG1.Image = null;
        view.cartaButtataG2.Image = null;
        view.briscolaGui.Image = null;
        MessageBox.Show("Cambio turno a giocatore "
        + corrente.ID);
        //aggiorno view settaCarteGui(); finePartita();
}
//se invece il mazzo contiene ancora carte
else
{
        //prende una carta dal mazzo il giocatore
        corrente daPrendere = corrente.prendiUnaCarta();
        //prende una carta dal mazzo il giocatore che ha
        // perso nel turno precedente
        correnteP = partita.tavolo.perdente;
        daPrendere = correnteP.prendiUnaCarta();
        //aggiorno view
        view.cartaButtataG2.Image = null;
        view.cartaButtataG1.Image = null;
}
//aggiorno view passando ai label i punteggi
// dei giocatori
view.label3.Text = "" + partita.g1.getPunteggio();
view.label4.Text = "" + partita.g2.getPunteggio();
//setto giocatore corrente
corrente = partita.getGCorrente(corrente, correnteP);
//setto le immagini da visualizzare aggiornando il view
SetCartaResources(view.carta1G2, retr);
```

```
                SetCartaResources(view.carta2G2, retr);
                SetCartaResources(view.carta3G2, retr);
                MessageBox.Show("Cambio turno a giocatore " + corrente.ID);
                settaCarteGui();
            }
            //se ancora si sta tirando per primo
            else if (partita.tavolo.numTurno == 0)
            {
                //setto il giocatore corrente della partita
                corrente = partita.getGCorrente(partita.g1, partita.g2);
                //butto la carta selezionata corrente.buttaCarta();
                //inverto il turno del giocatore
                partita.invertiTurnoG(partita.g1, partita.g2);
                //setto nuovamente il giocatore corrente della partita
                corrente = partita.getGCorrente(partita.g1, partita.g2);
                //visualizza e aggiorna view
                SetCartaResources(view.carta1G2, retr);
                SetCartaResources(view.carta2G2, retr);
                SetCartaResources(view.carta3G2, retr);
                MessageBox.Show("Cambio turno a giocatore " + corrente.ID);
                settaCarteGui();
            }
        }
};
view.carta1G2.MouseClick += (s, es) =>
{
        if (corrente == partita.g2)
        {
            //scelgo la carta rappresentata dalla picturebox daButtare =
            corrente.scegliCarta(corrente.listaGiocatore[0]);
            //aggiorno il view
            view.cartaButtataG2.Image = view.carta1G2.Image;
            view.carta1G2.Image = null;
            //se ha già tirato il primo giocatore
            if (partita.tavolo.numTurno == 1)
            {
                //butto la carta selezionata e partirà il confronto
```

```
corrente.buttaCarta();
// setto adesso il giocatore corrente
corrente = partita.getGCorrente(partita.tavolo.vincitore,
partita.tavolo.perdente);
//se il mazzo non ha piø carte
if (partita.mazzo.listaMazzo.Count == 0)
{
        //aggiorno view
        view.carta1G2.Image = null;
        view.cartaButtataG1.Image = null;
        view.cartaButtataG2.Image = null;
        view.briscolaGui.Image = null;
        MessageBox.Show("Cambio turno a giocatore "
        +corrente.ID);
        //Addormento il thread per 3 secondi dando quindi
        //il tempo all'avversario per
        //posizionarsi davanti lo schermo senza
        //guardare le carte dell'altro giocatore
        //aggiorno view
        settaCarteGui();
        finePartita();
}
//se invece il mazzo contiene ancora carte e non è quindi
else
{
        //prende una carta dal mazzo il giocatore corrente
        daPrendere = corrente.prendiUnaCarta();
        //prende una carta dal mazzo il giocatore che ha
        //perso nel turno precedente
        correnteP = partita.tavolo.perdente; daPrendere =
        correnteP.prendiUnaCarta();
        //aggiorno view view.cartaButtataG2.Image = null;
        view.cartaButtataG1.Image = null;
}
//aggiorno view passando ai label i punteggi
//dei giocatori
view.label3.Text = "" + partita.g1.getPunteggio();
```

```
                    view.label4.Text = "" + partita.g2.getPunteggio();
                    //setto giocatore corrente
                    corrente = partita.getGCorrente(corrente, correnteP);
                    //setto le immagini da visualizzare aggiornando il view
                    SetCartaResources(view.carta1G2, retr);
                    SetCartaResources(view.carta2G2, retr);
                    SetCartaResources(view.carta3G2, retr);
                    MessageBox.Show("Cambio turno a giocatore " + corrente.ID);
                    settaCarteGui();
                }
                //se ancora si sta tirando per primo
                else if (partita.tavolo.numTurno == 0)
                {
                    //setto il giocatore corrente della partita
                    corrente = partita.getGCorrente(partita.g1, partita.g2);
                    //butto la carta selezionata corrente.buttaCarta();
                    //inverto il turno del giocatore
                    partita.invertiTurnoG(partita.g1, partita.g2);
                    //setto nuovamente il giocatore corrente della partita
                    corrente = partita.getGCorrente(partita.g1, partita.g2);
                    //visualizza e aggiorna view
                    SetCartaResources(view.carta1G2, retr);
                    SetCartaResources(view.carta2G2, retr);
                    SetCartaResources(view.carta3G2, retr);
                    MessageBox.Show("Cambio turno a giocatore " + corrente.ID);
                    settaCarteGui();
                }
        }
};
view.carta3G1.MouseClick += (s, es) =>
{
        if (corrente == partita.g1)
{
        //scelgo la carta rappresentata dalla picturebox daButtare =
        corrente.scegliCarta(corrente.listaGiocatore[2]);
        //aggiorno il view
        view.cartaButtataG1.Image = view.carta3G1.Image;
```

```
view.carta3G1.Image = null;
//System.Threading.Thread.Sleep(3000);
//se ha già tirato il primo giocatore
if (partita.tavolo.numTurno == 1)
{
        //butto la carta selezionata e partirà il confronto
        corrente.buttaCarta();
        // setto adesso il giocatore corrente
        corrente = partita.getGCorrente(partita.tavolo.vincitore,
        partita.tavolo.perdente);
        //se il mazzo non ha più carte
        if (partita.mazzo.listaMazzo.Count == 0)
        {
                //aggiorno view
                view.carta3G1.Image = null;
                view.cartaButtataG1.Image = null;
                view.cartaButtataG2.Image = null;
                view.briscolaGui.Image = null;
                MessageBox.Show("Cambio turno a giocatore "
                + corrente.ID);
                //Addormento il thread per 3 secondi dando
                // quindi il tempo all'avversario per
                //posizionarsi davanti lo schermo senza guardare
                // le carte dell'altro giocatore
                //aggiorno view
                settaCarteGui(); finePartita();
        }
        //se invece il mazzo contiene ancora carte
        else
        {
                //prende una carta dal mazzo il giocatore corrente
                daPrendere = corrente.prendiUnaCarta();
                //prende una carta dal mazzo il giocatore
                // che ha perso nel turno precedente
                correnteP = partita.tavolo.perdente;
                daPrendere = correnteP.prendiUnaCarta();
```

```
                    //aggiorno view view.cartaButtataG2.Image = null;
                    view.cartaButtataG1.Image = null;
                }
                //aggiorno view passando ai label i punteggi
                //dei giocatori
                view.label3.Text = "" + partita.g1.getPunteggio();
                view.label4.Text = "" + partita.g2.getPunteggio();
                //setto giocatore corrente
                corrente = partita.getGCorrente(corrente, correnteP);
                //setto le immagini da visualizzare aggiornando il view
                SetCartaResources(view.carta1G1, retr);
                SetCartaResources(view.carta2G1, retr);
                SetCartaResources(view.carta3G1, retr);
                MessageBox.Show("Cambio turno a giocatore " + corrente.ID
                settaCarteGui();
        }
        //se ancora si sta tirando per primo
        else if (partita.tavolo.numTurno == 0)
        {
                //setto il giocatore corrente della partita
                corrente = partita.getGCorrente(partita.g1, partita.g2);
                //butto la carta selezionata corrente.buttaCarta();
                //inverto il turno del giocatore
                partita.invertiTurnoG(partita.g1, partita.g2);
                //setto nuovamente il giocatore corrente della partita
                corrente = partita.getGCorrente(partita.g1, partita.g2);
                //visualizza e aggiorna view
                SetCartaResources(view.carta1G1, retr);
                SetCartaResources(view.carta2G1, retr);
                SetCartaResources(view.carta3G1, retr);
                MessageBox.Show("Cambio turno a giocatore " + corrente.ID);
                settaCarteGui();
                }
        }
};
view.carta2G1.MouseClick += (s, es) =>
{
```

```
if (corrente == partita.g1)
{
        //scelgo la carta rappresentata dalla picturebox
        daButtare = corrente.scegliCarta(corrente.listaGiocatore[1]);
        //aggiorno il view
        view.cartaButtataG1.Image = view.carta2G1.Image;
        view.carta2G1.Image = null;
        //se ha già tirato il primo giocatore
        if (partita.tavolo.numTurno == 1)
        {
                //butto la carta selezionata e partirà il confronto
                corrente.buttaCarta();
                // setto adesso il giocatore corrente
                corrente = partita.getGCorrente(partita.tavolo.vincitore,
                partita.tavolo.perdente);
                //se il mazzo non ha più carte
                if (partita.mazzo.listaMazzo.Count == 0)
                {
                        //aggiorno view
                        view.carta2G1.Image = null;
                        view.cartaButtataG1.Image = null;
                        view.cartaButtataG2.Image = null;
                        view.briscolaGui.Image = null;
                        MessageBox.Show("Cambio turno a giocatore "
                        + corrente.ID);
                        //aggiorno view
                        settaCarteGui(); finePartita();
                }
                //se invece il mazzo contiene ancora carte
                else
                {
                        //prende una carta dal mazzo il giocatore corrente
                        daPrendere = corrente.prendiUnaCarta();
                        //prende una carta dal mazzo il giocatore che ha
                        //perso nel turno precedente
                        correnteP = partita.tavolo.perdente;
                        daPrendere = correnteP.prendiUnaCarta();
```

```
                        //aggiorno view
                        view.cartaButtataG2.Image = null;
                        view.cartaButtataG1.Image = null;
                    }
                //aggiorno view passando ai label i punteggi
                // dei giocatori
                view.label3.Text = "" + partita.g1.getPunteggio();
                view.label4.Text = "" + partita.g2.getPunteggio();
                //setto giocatore corrente
                corrente = partita.getGCorrente(corrente, correnteP);
                //setto le immagini da visualizzare aggiornando il view
                SetCartaResources(view.carta1G1, retr);
                SetCartaResources(view.carta2G1, retr);
                SetCartaResources(view.carta3G1, retr);
                MessageBox.Show("Cambio turno a giocatore " + corrente.ID);
                settaCarteGui();
        }
        //se ancora si sta tirando per primo
        else if (partita.tavolo.numTurno == 0)
        {
                //setto il giocatore corrente della partita
                corrente = partita.getGCorrente(partita.g1, partita.g2);
                //butto la carta selezionata corrente.buttaCarta();
                //inverto il turno del giocatore
                partita.invertiTurnoG(partita.g1, partita.g2);
                //setto nuovamente il giocatore corrente della partita
                corrente = partita.getGCorrente(partita.g1, partita.g2);
                //visualizza e aggiorna view
                SetCartaResources(view.carta1G1, retr);
                SetCartaResources(view.carta2G1, retr);
                SetCartaResources(view.carta3G1, retr);
                MessageBox.Show("Cambio turno a giocatore " + corrente.ID);
                settaCarteGui();
                }
        }
};
view.carta1G1.MouseClick += (s, es) =>
```

```
{
        if (corrente == partita.g1)
        {
                //scelgo la carta rappresentata dalla picturebox
                daButtare = corrente.scegliCarta(corrente.listaGiocatore[0]);
                //aggiorno il view
                view.cartaButtataG1.Image = view.carta1G1.Image;
                view.carta1G1.Image = null;
                //se ha già tirato il primo giocatore
                if (partita.tavolo.numTurno == 1)
                {
                        //butto la carta selezionata e partirà il confronto
                        corrente.buttaCarta();
                        //setto adesso il giocatore corrente
                        corrente = partita.getGCorrente(partita.tavolo.vincitore,
                        partita.tavolo.perdente);
                        //se il mazzo non ha piø carte
                        if (partita.mazzo.listaMazzo.Count == 0)
                        {
                                //aggiorno view view.carta1G1.Image = null;
                                view.cartaButtataG1.Image = null;
                                view.cartaButtataG2.Image = null;
                                view.briscolaGui.Image = null;
                                MessageBox.Show("Cambio turno a giocatore " +
                                corrente.ID);
                                //aggiorno view
                                settaCarteGui();
                                finePartita();
                        }
                        //se invece il mazzo contiene ancora carte
                        Else
                         {
                                //prende una carta dal mazzo il giocatore corrente
                                daPrendere = corrente.prendiUnaCarta();
                                //prende una carta dal mazzo il giocatore che
                                // ha perso nel turno precedente
                                correnteP = partita.tavolo.perdente;
```

```
                    daPrendere = correnteP.prendiUnaCarta();
                    //aggiorno view
                    view.cartaButtataG2.Image = null;
                    view.cartaButtataG1.Image = null;
                }
                //aggiorno view passando ai label i punteggi dei giocatori
                view.label3.Text = "" + partita.g1.getPunteggio();
                view.label4.Text = "" + partita.g2.getPunteggio();
                //setto giocatore corrente
                corrente = partita.getGCorrente(corrente, correnteP);
                //setto le immagini da visualizzare aggiornando il view
                SetCartaResources(view.carta1G1, retr);
                SetCartaResources(view.carta2G1, retr);
                SetCartaResources(view.carta3G1, retr);
                MessageBox.Show("Cambio turno a giocatore "
                + corrente.ID);
                settaCarteGui();
            }
        //se ancora si sta tirando per primo else if (partita.tavolo.numTurno == 0)
        {
            //setto il giocatore corrente della partita
            corrente = partita.getGCorrente(partita.g1, partita.g2);
            //butto la carta selezionata corrente.buttaCarta();
            //inverto il turno del giocatore
            partita.invertiTurnoG(partita.g1, partita.g2);
            //setto nuovamente il giocatore corrente della partita
            corrente = partita.getGCorrente(partita.g1, partita.g2);
            //visualizza e aggiorna view
            SetCartaResources(view.carta1G1, retr);
            SetCartaResources(view.carta2G1, retr);
            SetCartaResources(view.carta3G1, retr);
            MessageBox.Show("Cambio turno a giocatore "
            + corrente.ID);
            settaCarteGui();
            }
        }
}
```

```
public void finePartita()
{
        fine = new Fine();
        //se i due giocatori non hanno più carte disponibili nei propri mazzi
        SetCartaResources(fine.pictureBox1, "coppa");
        if (partita.g1.listaGiocatore.Count == 0 && partita.g2.listaGiocatore.Count == 0)
        {
                String vincSt = null;
                //stringa che salva il punteggio
                //del vincitore
                String perdSt = null;
                //stringa che salva il punteggio del perdente
                //se il punteggio del g1 è > del punteggio di g2 allora la partita
                //sarà vinta da g1
                if (partita.g1.punteggio > partita.g2.punteggio)
                {
                        vincSt = " " + partita.g1.punteggio; perdSt = " " +
                        partita.g2.punteggio;
                        MessageBox.Show("FINE PARTITA. Vince G1"
                        + vincSt + " a " + perdSt);
                        //mostro la finestra di fine partita
                        fine.Show();
                        view.Close();
                }
                //se il punteggio del g2 è > del punteggio di g1 allora la partita
                //sarà vinta da g2
                else if (partita.g1.punteggio < partita.g2.punteggio)
                {
                        vincSt = " " + partita.g2.punteggio; perdSt = " " +
                        partita.g1.punteggio;
                        MessageBox.Show("FINE PARTITA. Vince G2"
                        + vincSt + " a " + perdSt);
                        fine.Show();
                        view.Close();
                }
                //se il punteggio del g1 è = al punteggio di g2 allora
                // la partita non sarà vinta da nessuno
```

```
            else if (partita.g1.punteggio == partita.g2.punteggio)
            {
                vincSt = " " + partita.g2.punteggio;
                perdSt = " " + partita.g1.punteggio;
                MessageBox.Show("FINE PARTITA.");
                fine.label1.Text = "FINE PARTITA. Vince l'amicizia";
                SetCartaResources(fine.pictureBox1, "amicizia");
                fine.Show();
                view.Close();
            }
        }
}
```

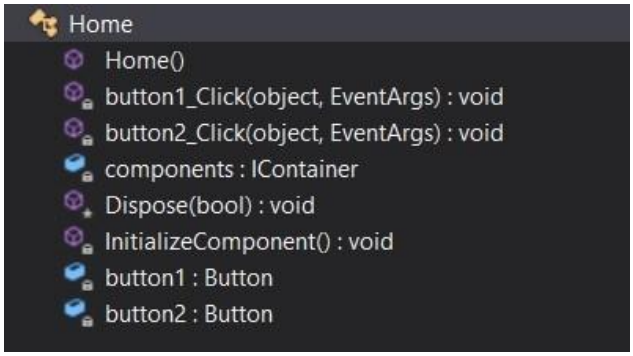
4.9 Home

La home rappresenta il primo form che si apre all'avvio dell'applicazione e consiste in:

- 2 button che permettono il primo di avviare la partita e il secondo di uscire dal gioco.

Esiste un solo metodo costruttore:

- Home().

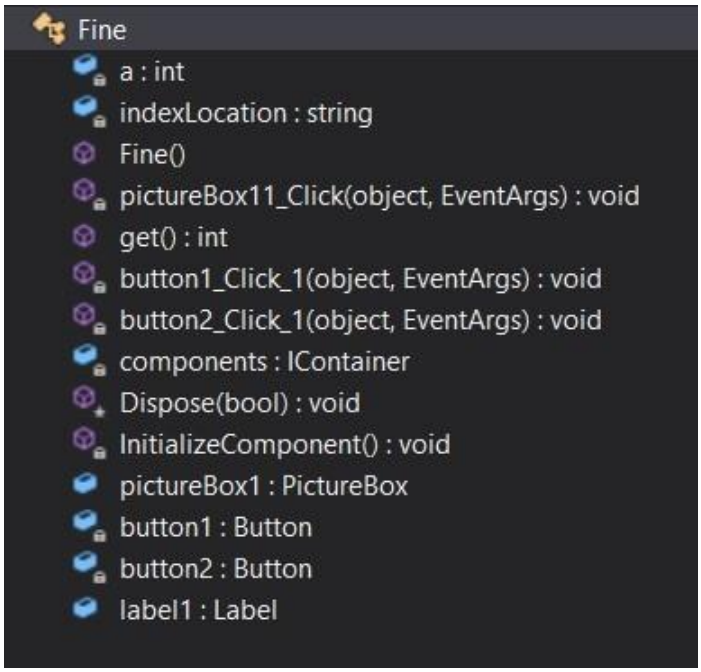



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace briscolax2
    {
        public partial class Home : Form
        {
            public Home()
```

```
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            LogicaCarte.Partita partita = new LogicaCarte.Partita();
            partita.inizializzaPartita();
            View view = new View();
            //dichiaro il control
            Control control = new Control(view, partita);
            control.settaControlli();
            control.settaCarteGui();
            view.Show();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

4.10 Fine

Fine è invece un Form composto da una pictureBox la quale immagine dipende dal risultato della partita.
La stessa cosa si pu dire per l'uso del label sotto la pictureBox e dei due button.
Il primo per avviare la rivincita, quindi inizializzare un'altra partita e il secondo per uscire dal gioco.




```
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;
```

```
namespace briscolax2
{
    public partial class Fine : Form
    {
        public Fine()
        {
            InitializeComponent(); }
            private void pictureBox11_Click(object sender, EventArgs e)
            {
            }
            public int get()
            {
                if (a == 1)
                {
                    return a;
                }
                else
                {
                    return 0;
                }
            }
            private void button1_Click_1(object sender, EventArgs e)
            {
```

```
                this.Visible = false;
                LogicaGioco.Partita partita = new LogicaGioco.Partita();
                partita.inizializzaPartita();
                View view = new View();
                //dichiaro il control
                Control control = new Control(view, partita);
                control.settaControlli();
                control.settaCarteGui();
                view.Show();
            }
            private void button2_Click_1(object sender, EventArgs e)
            {
                this.Close();
            }
        }
}
```

4.11 Program

La classe program da cui si avvia il progetto con l'interaccia creata è dentro la soluzione briscolax2.

```
namespace briscolax2
        {
        static class Program
        {
                // <summary>
                // Punto di ingresso principale dell'applicazione.
                // </summary> [STAThread]
                static void Main()
                {
                        Application.SetCompatibleTextRenderingDefault(false);
                        //dichiaro la view
                        Home home = new Home();
                        Application.EnableVisualStyles();

                        Application.Run(home);

                }
        }
}
```

5 Compilazione ed esecuzione

L'ambiente utilizzato per lo sviluppo dell'applicazione è Visual Studio 2017 Enterprise - framework .NET 4.5.2.
Per l'esportazione dei diagrammi uml è stato utilizzato anche Visual Studio 2015 Professional.
Per avviare l'applicazione è prima importante compilarla, quindi:

- Estrarre la cartella briscolax2 dal file .rar briscolax2.rar
- Doppio click sul le con estensione .sln (briscolax2.sln) che si trova dentro la cartella briscolax2 ed attendere il caricamento del progetto.
- Compilare la soluzione.
- Infine andare sulla cartella Release dell'applicazione e fare partire il file eseguibile.

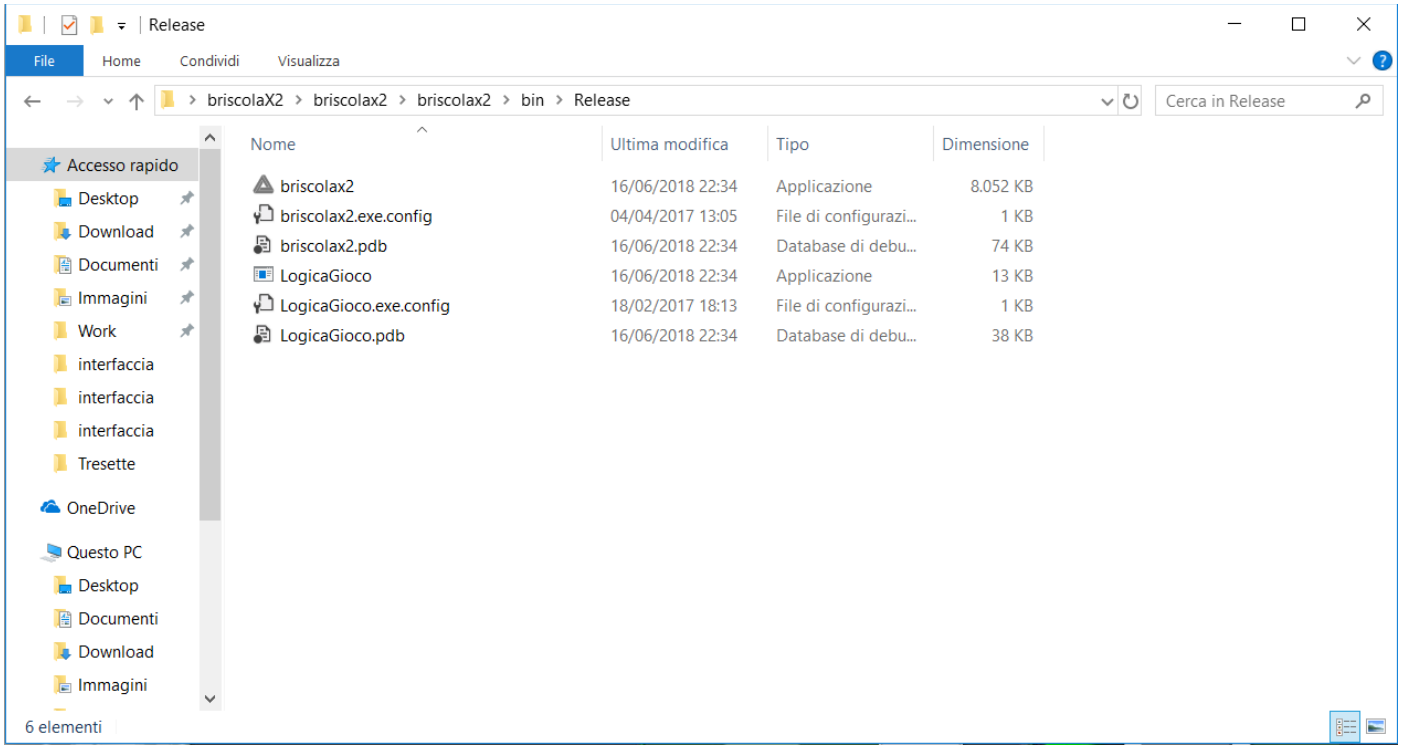
Per l'uso dell'applicazione è consigliato che il sistema utilizzi Windows con framework 4.5.2 o superiori.
Il programma è stato testato in macchine con architetture diverse mostrate di seguito in tabella:

SISTEMA OPERATIVO	PROCESSORE	RAM	ARCHITETTURA
Windows 10	Intel® Core™ i7-6700 HQ CPU @2.60 GHz	8 GB	64 bit
Windows 10	Intel® Core ™ i5-5200 U CPU @ 2.30 GHz	8 GB	64 bit
Windows10	Intel Atom Inside CPU @1.60 GHz	2 GB	32 bit
Windows 10	Intel® Core™ i7-4720 MQ CPU @2.20 GHz	16 GB	64 bit
Windows 10	Intel® Core™ i7-6700 HQ CPU @2.60 GHz	16 GB	64 bit

6 Testing

Per dimostrare il funzionamento dell'applicazione sono stati inseriti degli screenshot riguardanti una nuova partita.

Per prima cosa bisogna avviare il file briscolax2.exe dalla cartella release.



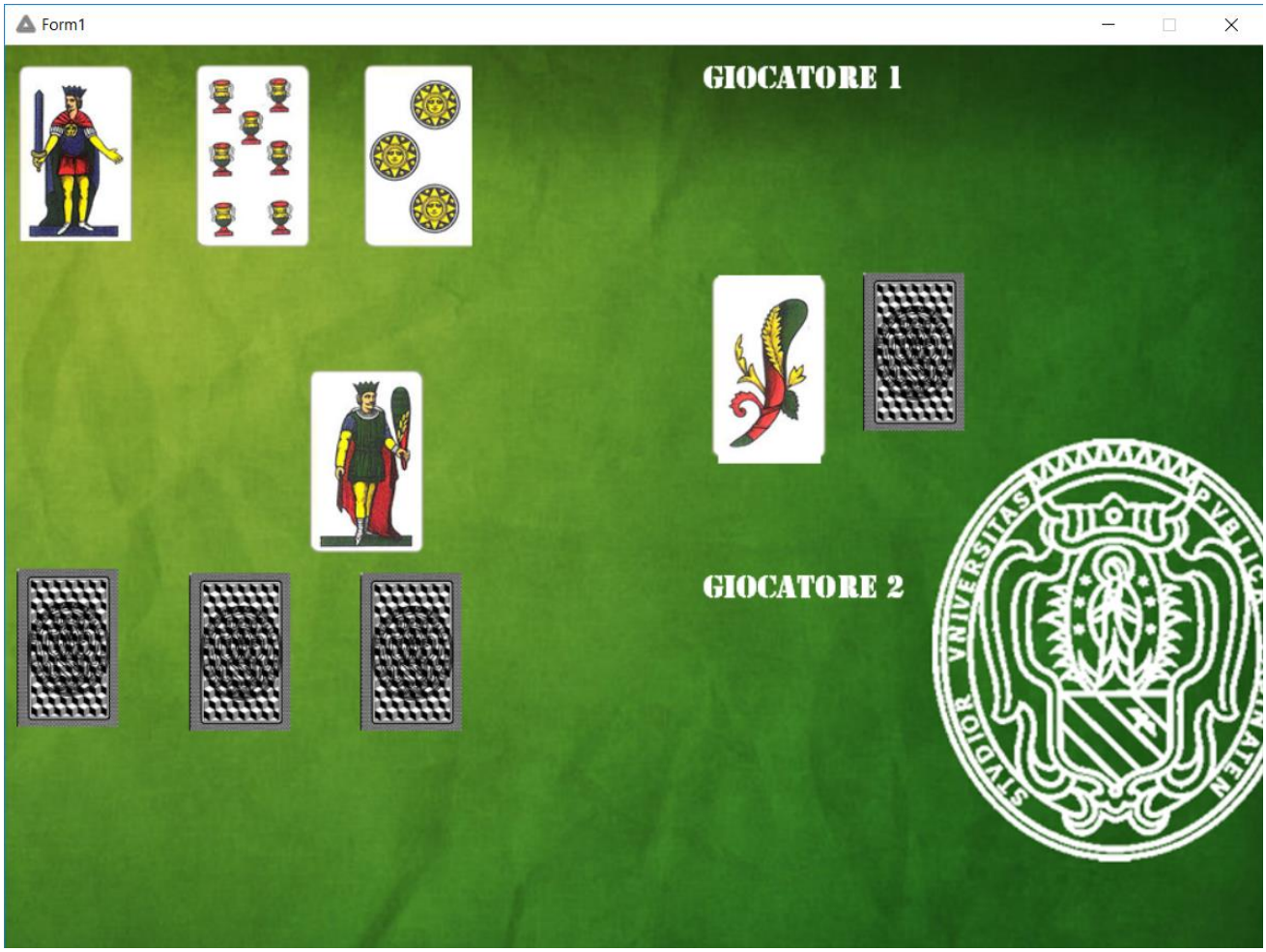
Apparirà quindi la schermata principale del gioco.

Sarà possibile giocare, visualizzare le regole/istruzioni del gioco nonché chiudere l'applicazione:

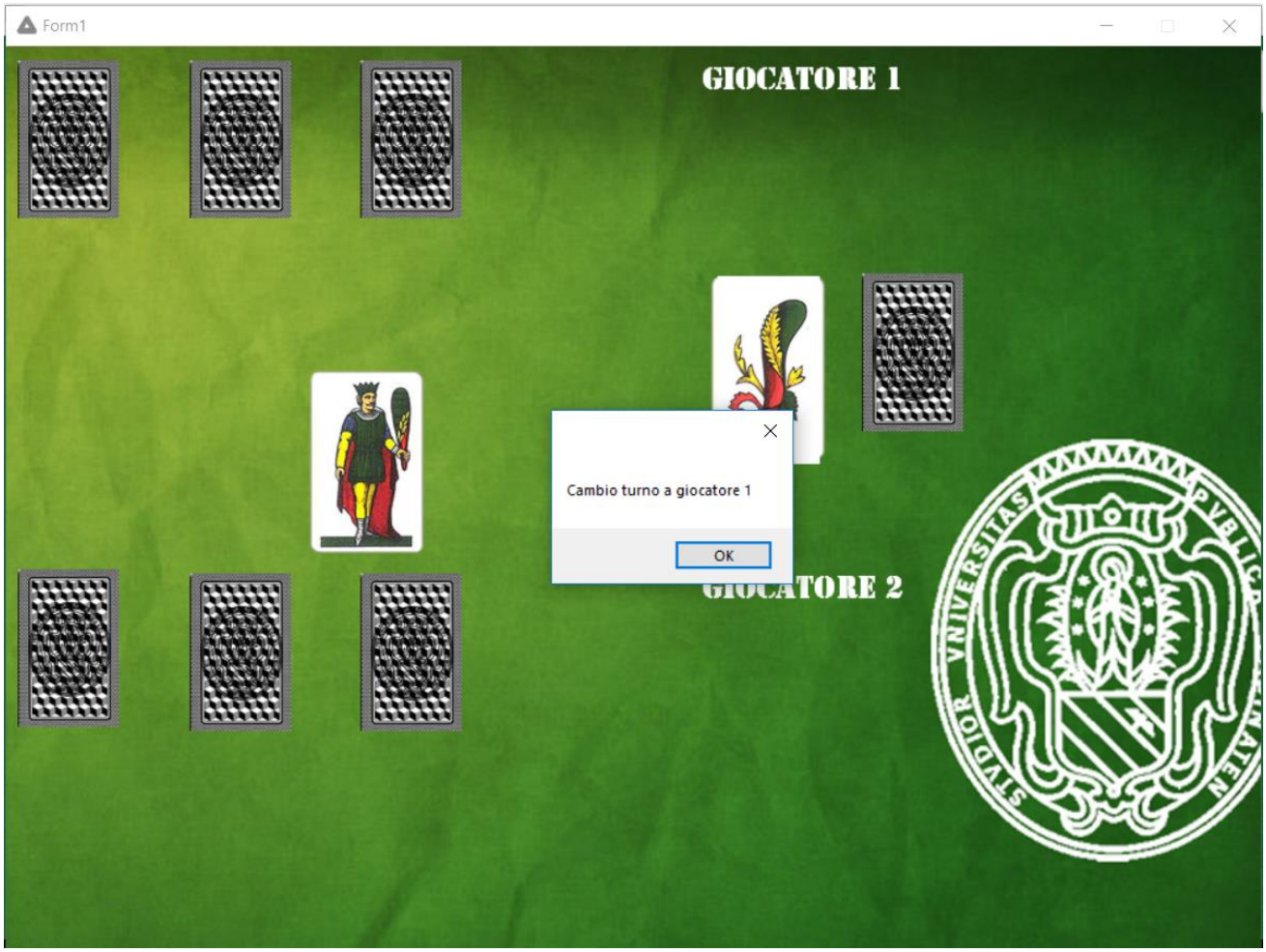
- form contenente le istruzioni del gioco


Istruzioni
Si gioca con un mazzo di 40 carte diviso in 4 semi di 10 carte ciascuno. I punti disponibili per ciascun gioco sono in totale 120, vince chi ne realizza almeno 61; se i punti sono 60 il gioco è pari. il gioco classico segue delle regole basilari sostanzialmente semplici: ogni giocatore si ritrova 3 carte in mano, mentre la prima del mazzo, viene scoperta e posta sul tavolo per designare il seme di briscola; per convenzione la carta che designa la briscola, viene messa sotto il mazzo per tutta la durata della partita, cosicché sarà sempre visibile ai giocatori e sarà l'ultima carta ad essere pescata, dopodiché iniziano una serie di mani.
Ogni giocatore giocherà la sua carta che riterrà più opportuna, ovvero la poserà fisicamente sul tavolo, con lo scopo di aggiudicarsi la mano o di totalizzare il maggior numero di punti. Da parte dei giocatori non esiste alcun obbligo di giocare un particolare tipo di seme.
L'aggiudicazione della mano avviene secondo regole molto semplici:
il primo giocatore di mano, così viene chiamato colui che posa la prima carta a ogni giro, determina il seme di mano o dominante e virtualmente è il vincitore temporaneo della mano.
La mano può essere temporaneamente aggiudicata da un altro giocatore se posa una carta del seme di mano, ma con valore di presa maggiore, oppure giocando una qualsiasi carta del seme di briscola, anche con valore di presa inferiore.
Alla fine la mano viene aggiudicata dal giocatore che ha posato la carta di briscola col valore di presa maggiore, o, in mancanza, da colui che ha giocato la carta del seme di mano comunque col valore di presa maggiore; e infine, in mancanza di carte di briscola o altre carte del seme di mano, dal primo giocatore della mano, in quanto rimane l'unico ad aver giocato la carta del seme di mano.
Si badi che il seme di mano, essendo determinato di volta in volta dalla prima carta giocata nella mano, può anche incidentalmente coincidere col seme di briscola; in questo caso la mano se la aggiudica colui che ha giocato la briscola maggiore.
Il vincitore della mano successivamente sarà il primo a prendere la prima carta del tallone, seguito dall'altro giocatore e sarà sempre il primo ad aprire la mano successiva, e quindi a decidere il nuovo seme di mano.

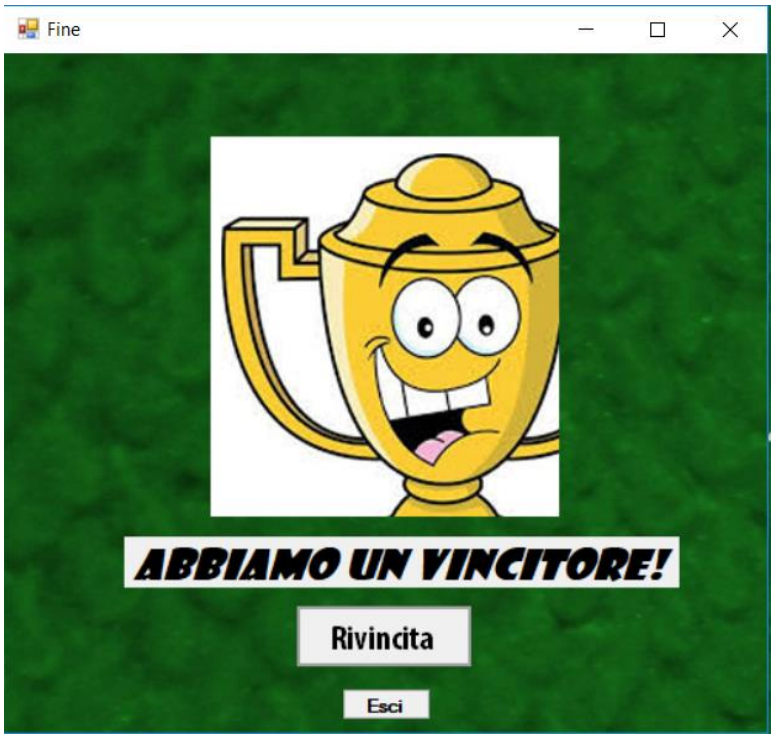
- form del gameplay:



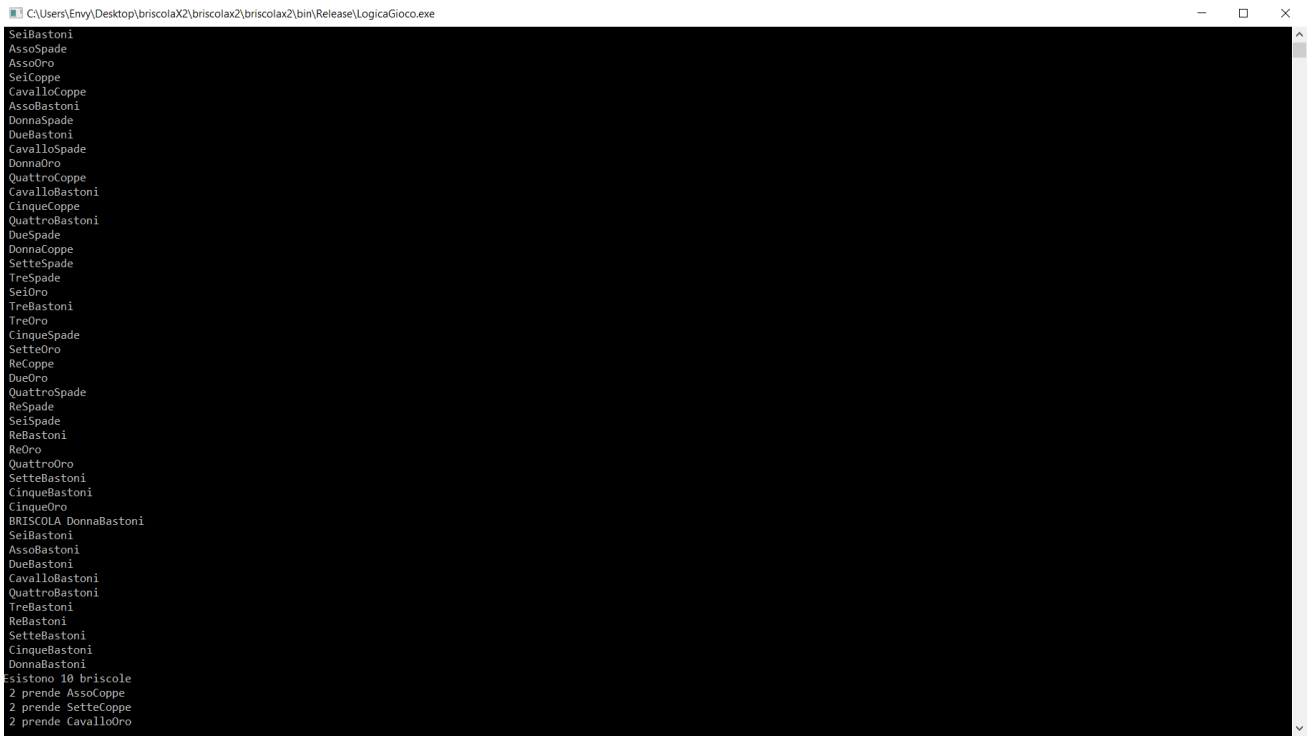

Form1
GIOCATORE 1
GIOCATORE 2


Form1
GIOCATORE 1
Cambio turno a giocatore 1
OK
GIOCATORE 2

In caso di vittoria di un giocatore appare la seguente schermata:



Segue una schermata di testing del programma:



Strumenti utilizzati

Per la realizzazione dell'intera applicazione è stato utilizzato il seguente parco software:

- VisualStudio 2017 Enterprise
- Notepad++

Per la realizzazione della relazione invece si è fatto uso di:

- Microsoft Word Student Edition
- VisualStudio 2015 per la creazione automatica di diagrammi UML delle classi.

Il sistema operativo utilizzato è Windows 10 Home Creator Update con la seguente configurazione: