Analisi Tecnica

Mattia Zanini 4F

Pre-partita

All'avvio il programma crea l'array che rappresenta le carte del memory, per far ciò, vengono inizializzati:

- un array contenente i valori delle 4 coppie

```
string[] fiori = new string[] { "magnolia", "margherita", "primula", "campane blu" };
```

- delle variabili `int` per tener traccia di quante volte sono state sorteggiati i valori dell'array `fiori`

```
int nGenmagnolia = 0, nGenMargherita = 0, nGenPrimula = 0, nGenCampane_Blu = 0;
```

- l'array che conterrà le 4 coppie

```
string[] fioriCarte = new string[8];
```

Inoltre viene creato l'oggetto della classe `Random` per poter utilizzare la sua funzione `Next()`

```
Random genCarte = new Random();
```

Dopo di ciò, il programma riempie con un ciclo **for** l'array `fioriCarte`

```
for (int i = 0; i < 8; i++)
{
  string fioreGenerato = genCarteRandom(fiori, genCarte, ref nGenmagnolia,
  ref nGenMargherita ref nGenPrimula, ref nGenCampane_Blu);

  if (i > 1 && controlloCarte(fioriCarte, ref fioreGenerato, ref nGenmagnolia,
  ref nGenMargherita, ref nGenPrimula, ref nGenCampane_Blu) == -1)
  {
    i--;
  }
  else
  {
    fioriCarte[i] = fioreGenerato;
  }
}
```

La funzione `genCarteRandom()` si occupa di restituire un valore, ricavato in modo randomico tramite la funzione `Next()` dall'array fiori

```
private string genCarteRandom(string[] arrayFiori, Random gnCard, ref int
nGenmagnolia,
ref int nGenMargherita, ref int nGenPrimula, ref int nGenCampane_Blu)
{
  string fioreGenerato = "";
  int posGenerata = 0;
  posGenerata = gnCard.Next(0,4);
  fioreGenerato = arrayFiori[posGenerata];
  switch (fioreGenerato)
  {
    case "magnolia":
      nGenmagnolia = aumentoNumFioreGenerato(nGenmagnolia);
      break;
    case "margherita":
      nGenMargherita = aumentoNumFioreGenerato(nGenMargherita);
      break;
    case "primula":
      nGenPrimula = aumentoNumFioreGenerato(nGenPrimula);
      break;
    case "campane blu":
      nGenCampane_Blu = aumentoNumFioreGenerato(nGenCampane_Blu);
    break;
  }
  return fioreGenerato;
}
```

`gnCard.Next(0,4)` restituisce un valore compreso tra 0 e 3 (inclusi) e tramite questo si ricava da `arrayFiori` un valore e con uno switch, a seconda di cosa si è ottenuto si aumenta il valore che rappresenta il quantitativo di sorteggi che un determinato valore ha avuto

```
private int aumentoNumFioreGenerato(int genNum)
{
  if (genNum < 3)
  {
    genNum++;
  }
  return genNum;
}
```

Dopo che si è ottenuto randomicamente un valore per la carta, si controlla se è possibile la sua generazione, ovvero se non sono presenti più di 2 carte con lo stesso valore (sennò non rispetterebbe le regole del gioco "formare delle coppie di carte").
Per fare il controllo viene utilizzato un **if** che partirà ad eseguire i controllo dal terzo ciclo del **for**, la funzione `controlloCarte()` si occuperà del problema.

```
private int controlloCarte(string[] arrayCard, ref string fioreGenerato,
ref int nGenmagnolia, ref int nGenMargherita, ref int nGenPrimula, ref int
nGenCampane_Blu)
{
  arrayCard = arrayCard.Where(c => c != null).ToArray();
  for(int i = 0; i < arrayCard.Length; i++)
  {
    if(arrayCard[i] == fioreGenerato)
    {
      switch (fioreGenerato)
      {
        case "magnolia":
          if(nGenmagnolia == 3)
          {
            return -1;
          }
          break;
        case "margherita":
          if (nGenMargherita == 3)
          {
            return -1;
          }
          break;
        case "primula":
          if (nGenPrimula == 3)
          {
            return -1;
          }
          break;
        case "campane blu":
          if (nGenCampane_Blu == 3)
          {
            return -1;
          }
          break;
      }
    }
  }
  return 1;
}
```

Inanzitutto la funzione elimina i valori `null` dall'array, per evitare controlli inutili nel ciclo **for**, poi la funzione vede se il valore generato è già stato registrato all'interno dell'array `fioriCarte`, nel caso non sia presente restituisce `1` e permette l'assegnazione del valore nell'array principale, nel caso invece il valore generato è già presente, tramite uno switch, esamina quante volte è stato generato, se è stato generato per più di due volte (aumentoNumFioreGenerato() assegna al massimo a 3 il quantitativo, anche se esso lo supera) la funzione restituisce `-1` e quindi, la funzione che si occupare di riempire l'array `fioriCarte`, diminuisce il valore del contatore del **for** e ripete l'operazione da capo, se invece il valore è stato generato per meno di 2 volte, resituisce `1` e permette l'assegnazione

Una volta che il tutto si è concluso, vengono assegnate le immagini standard alle carte del form, con la funzione `assegnazioneSfondoCarte()` e successivamente ques'ultimo viene visualizzato

```
private void assegnazioneSfondoCarte()
{
  carta1.Image = Properties.Resources.sfondo_carta;
  carta2.Image = Properties.Resources.sfondo_carta;
  carta3.Image = Properties.Resources.sfondo_carta;
  carta4.Image = Properties.Resources.sfondo_carta;
  carta5.Image = Properties.Resources.sfondo_carta;
  carta6.Image = Properties.Resources.sfondo_carta;
  carta7.Image = Properties.Resources.sfondo_carta;
  carta8.Image = Properties.Resources.sfondo_carta;
}
```

Durante la partita

Quando il giocatore clicca su una qualsiasi carta, viene eseguita la funzione `_Click()` di questa che, a sua volta esegue la funzione `clickCarta()`. La funzione diminuisce di 1 il valore passato dalle carte (per far corrispondere la carta al corrispettivo valore nell'array), un if controlla se la determinata carta è stata già selezionata, nel caso non lo fosse,tramite uno switch si cambia l'immagine della carta, le si da lo stato di "selezionata" e si verifica di quale giocatore è il turno con l'ausilio della funzione `controlloTurno()`.

```
private void clickCarta(int numCarta)
{
  numCarta -= 1
  if(sceltaGiocatore[numCarta] == false)
  {
    switch (numCarta)
    {
      case 0:
        carta1.Image = cartasfondo(numCarta);
        sceltaGiocatore[numCarta] = true;
        controlloTurno(numCarta);
        break;
      case 1:
        carta2.Image = cartasfondo(numCarta);
        sceltaGiocatore[numCarta] = true;
        controlloTurno(numCarta);
        break;
      case 2:
        carta3.Image = cartasfondo(numCarta);
        sceltaGiocatore[numCarta] = true;
        controlloTurno(numCarta);
        break;
      case 3:
        carta4.Image = cartasfondo(numCarta);
        sceltaGiocatore[numCarta] = true;
        controlloTurno(numCarta);
        break;
```

```
      case 4:
        carta5.Image = cartasfondo(numCarta);
        sceltaGiocatore[numCarta] = true;
        controlloTurno(numCarta);
        break;
      case 5:
        carta6.Image = cartasfondo(numCarta);
        sceltaGiocatore[numCarta] = true;
        controlloTurno(numCarta);
        break;
      case 6:
        carta7.Image = cartasfondo(numCarta);
        sceltaGiocatore[numCarta] = true;
        controlloTurno(numCarta);
        break;
      case 7:
        carta8.Image = cartasfondo(numCarta);
        sceltaGiocatore[numCarta] = true;
        controlloTurno(numCarta);
        break;
      }
  }
}

private void controlloTurno(int nCard)
{
  carteSelezionate++;
  controlloCoppie(ref nCard);
}
```

`controlloTurno()` riceve il valore della carta come parametro, aumento il numero delle carte che sono state selezionate durante la partita e passa il parametro a sua volta ad una altra funzione `controlloCoppie()`

```
int card1, card2;
private void controlloCoppie(ref int nCard)
{
  if (carteSelezionate % 2 == 0)
  {
    card2 = nCard;
    if(fioriCarte[card1] != fioriCarte[card2])
    {
      sceltaGiocatore[card1] = false;
      sceltaGiocatore[card2] = false;
      wait(500);
      assegnazioneSfondoCarteNonCoppie(card1);
      assegnazioneSfondoCarteNonCoppie(card2);
      carteSelezionate -= 2;
      turno = !turno;
      if (turno)
      {
        this.BackColor = Color.Red;
      }
```

```
      else
      {
        this.BackColor = Color.Blue;
      }
    }
    else
    {
      if (turno)
      {
        coppieG2++;
      }
      else
      {
        coppieG1++;
      }
    }
  }
  else
  {
    card1 = nCard;
  }
  if(carteSelezionate == 8)
  {
    restartGame.Visible = true;
    this.BackColor = Color.White;
    if(coppieG1 > coppieG2)
    {
      MessageBox.Show($"La partita si è conclusa\nHa vinto il giocatore 1");
    }
    else if(coppieG2 > coppieG1)
    {
      MessageBox.Show($"La partita si è conclusa\nHa vinto il giocatore 2");
    }
    else
    {
      MessageBox.Show($"La partita si è conclusa\nPareggio");
    }
  }
}
```

Quando il numero delle carte selezionate è dispari, memorizza il valore della carta dispari cliccata, se invece il numero delle carte selezionate è positivo, salva il valore della carta positiva selezionata e successivamente controlla se nell'array gli elementi corrispondenti ai valori salvati delle due carte siano gli stessi, in questo caso, si aumenta il valore delle coppie formate dal giocatore che sta giocando in quel turno, invece nel caso in cui gli elementi siano diversi, si disattiva lo stato di "selezione" delle carte, il programma aspetta 0,5s, tramite la funzione `wait()`, poi riassegna alle due carte precedentemente selezionate l'immagine iniziale, si diminuisce il valore delle carte selezionate di 2, si cambia turno e colore dello sfondo della finestra. Se, quando viene chiamata la funzione, il numero delle carte selezionate è pari a 8, si mostra a schermo il vincitore o l'eventuale pareggio dei due giocatori, tramite una `MessageBox`.

```
public void wait(int milliseconds)
{
  var timer1 = new System.Windows.Forms.Timer();
  if (milliseconds == 0 || milliseconds < 0) return;

  timer1.Interval = milliseconds;
  timer1.Enabled = true;
  timer1.Start();

  timer1.Tick += (s, e) =>
  {
  timer1.Enabled = false;
  timer1.Stop();
  };

  while (timer1.Enabled)
  {
    Application.DoEvents();
  }
}
//P.S. preso da StackOverflow
```

`wait()` permette al programma di spospendersi per il tempo (millisecondi) passato in parametro, con un if, controlla se il tempo è minore o uguale a zero, se non lo è assegna alla variabile timer l'intervallo pari al valore ricevuto come parametro, viene attivato il timer e fatto partire e viene concluso dopo aver superato il valore assegnato al timer, in questo lasso di tempo il programma non può ricevere input