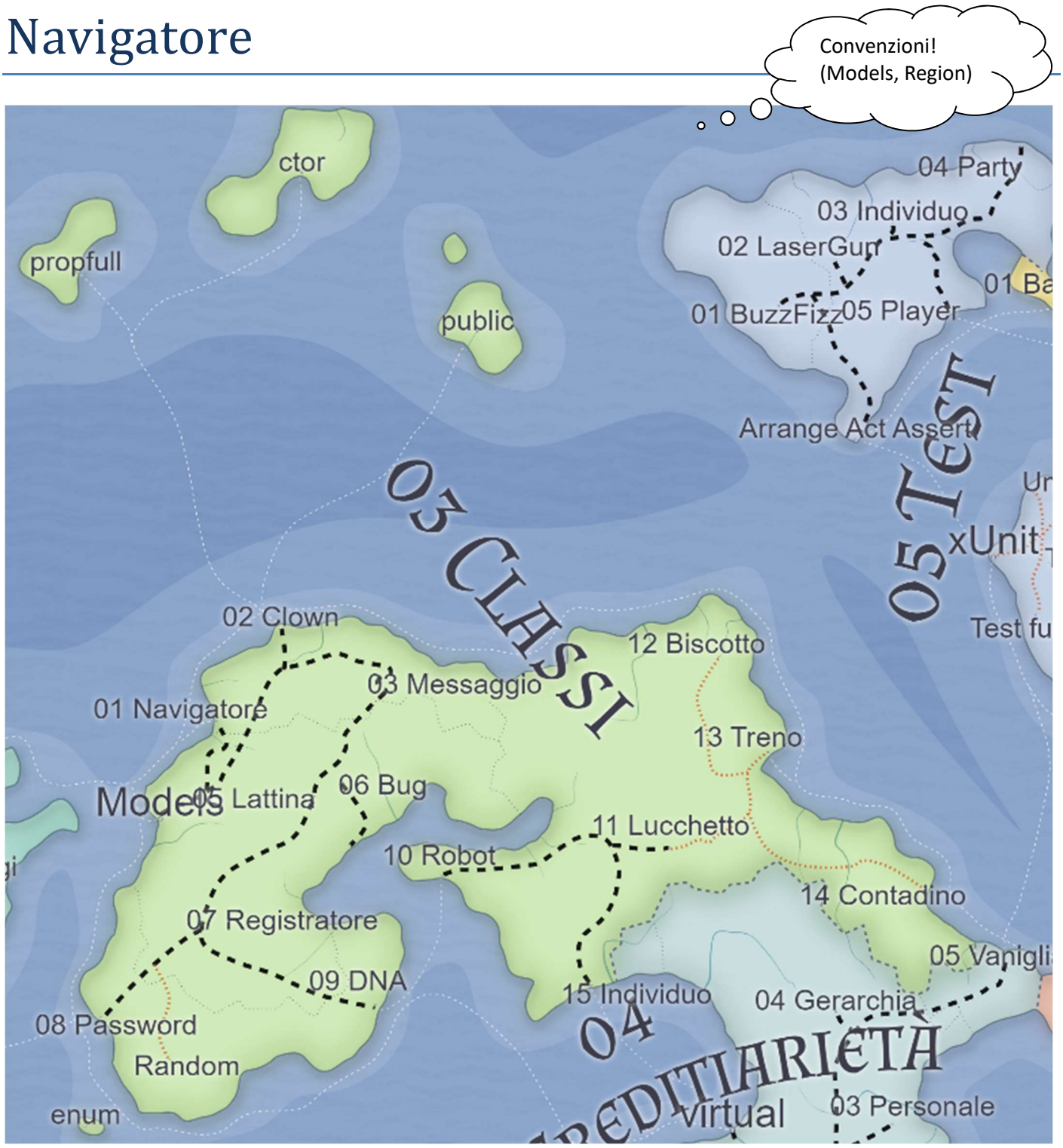
Navigatore


Convenzioni!
(Models, Region)
ctor
propfull
public
04 Party
03 Individuo
02 LaserGun
01 BuzzFizz
05 Player
Arrange Act Assert
05 Test
xUnit
03 CLASSI
02 Clown
12 Biscotto
03 Messaggio
01 Navigatore
13 Treno
Models
05 Lattina
06 Bug
11 Lucchetto
10 Robot
14 Contadino
07 Registratore
09 DNA
15 Individuo
04 Gerarchia
08 Password
Random
enum
04
virtual
03 Personale

Obiettivi

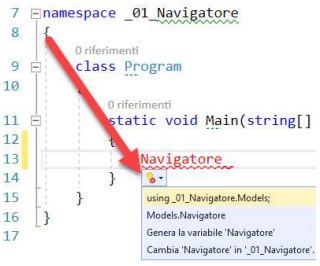
- Classi (public!)
- **Cartella Models**
- Proprietà
- **Region**
- SmartTag
- Abbreviazioni (ctor, propfull)
- Random
- Enum
- Librerie esterne (MahApps & HandyControls)

Teoria

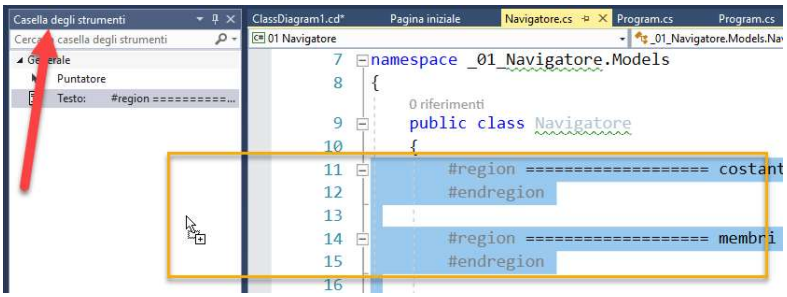
In informatica si deve spesso “tradurre” la realtà con del codice, semplificando molti aspetti in modo da utilizzare unicamente gli aspetti che interessano.

In questa fase, le classi che si creeranno vengono pure raggruppati in una cartella specifica: **Models**. I nomi delle classi non seguono una cnvezione particolare se non quello di avere dei nomi sensati alla loro funzionalità.

Quando si vorrà utilizzerare la classe, VS suggerirà, tramite uno smarttag, alcune varianti per risolvere l’errore (di solito inserendo lo using) che eseguirà dopo la scelta.



Per evitare di fare un copia/incolla della struttura della classe con le region, è possibile aggiungere la parte desiderata facendo un drag&drop del testo selezionato nella casella degli strumenti, rinominando l’elemento se del caso. Per poter utilizzare il codice, basta posizionare il cursore dove si vuole e poi eseguire un 2xclick dalla casella degli strumenti.



Esempio

03 Classi (Soluzione)
01 Navigatore (App Console)

Il pensiero di Mike a proposito del suo problema

Mike è un programmatore deve andare al colloquio per un nuovo posto. Non vede l'ora di mostrare le sue capacità di C#, ma prima deve arrivarci, ed è in ritardo.

- Mike si immagina quali vie percorrere per raggiungere il luogo (prendo A2 direzione Sud, esco a Mendrisio e poi vado su via Vignalunga – definisce la destinazione ottenendo il percorso)
- Per fortuna ha una radio che annuncia dei forti rallentamenti sulla A2 sul ponte di Melide. Mike ha una nuova informazione sulla strada da evitare.
- Mike sceglie una via alternativa (strada cantonale) per raggiungere il luogo dell'appuntamento in orario.

Il pensiero del navigatore di Mike a proposito del suo problema

Mike ha il proprio sistema di navigazione per girare il Ticino

```
InserireDestinazione("Via Vignalunga, 6850 Mendrisio");
string strada;
strada = TrovarePercorso();

"Prendere l'autostrada a Lugano Paradiso direzione sud, uscire a Mendrisio, …"
InserireDeviazione("A2 ponte diga di Melide");
string strada;
strada = TrovarePercorso();
```

"Prendere la strada cantonale a Lugano Paradiso direzione Melide, continuare sulla strada cantonale fino a Mendrisio, …"

Il sistema di navigazione di Mike risolve il problema allo stesso modo.

La classe navigatore ha metodi per definire e modificare le strade.

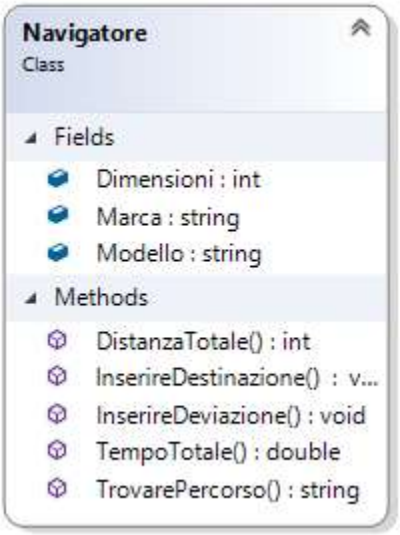
La classe ha metodi nelle quali avvengono le azioni. Ma a differenza del metodo button_Click(), essi sono predisposti per un unico problema: trovare percorsi in un paese. Per questo Mike li ha raccolti tutti in una classe chiamata Navigatore. Mike ha disegnato la sua classe Navigatore in modo da definire o modificare facilmente le strade. Per trovare un percorso, il programma chiama il metodo InserireDestinazione() e usa TrovarePercorso() per ottenerlo in una stringa. Se deve modificare il percorso, richiama InserireDeviazione() e poi richiama di nuovo TrovarePercorso() per ottenere le nuove indicazioni.

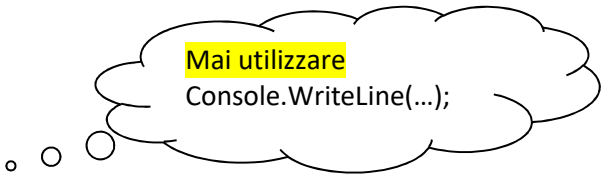
Navigatore
+Marca : string +Dimensioni : int +Modello : string
+InserireDestinazione() +TrovarePercorso() : string +InserireDeviazione() +DistanzaTotale() : int +TempoTotale() : double

Ogni metodo è costituito da una sequenza di comandi. Alcuni metodi eseguono solamente i comandi e poi terminano. Altri hanno un valore di ritorno (return) e cioè un valore che è stato calcolato o generato al suo interno e poi ritornato al comando che ha chiamato quel metodo. Il tipo di ritorno è chiamato tipo di ritorno.

Il comando return indica al metodo di uscire immediatamente. Se il metodo non ha un valore di ritorno (nell'intestazione si ha come tipo di ritorno void) return è seguito da un punto e virgola o è possibile tralasciarlo. Se invece ha un tipo di ritorno. Il comando è imperativo e deve essere seguito dal valore stesso (come valore o come variabile o come calcolo).

```
public class Navigatore
{
    #region =01=== costanti & statici ======================
    #endregion

    #region =02=== membri & proprietà ======================
    public string Marca;
    public int Dimensioni;
    public string Modello;
    #endregion

    #region =03=== costruttori =============================
    #endregion

    #region =04=== metodi private e aiuto ==================
    #endregion

    #region =05=== metodi public ===========================
    /// <summary>
    /// Definire il punto di arrivo
    /// </summary>
    /// <param name="arrivo">destinazione</param>
    public void InserireDestinazione(string arrivo)
    {

    }
    /// <summary>
    /// Ottenere il tragitto
    /// </summary>
    /// <returns>Percorso da effettuare</returns>
    public string TrovarePercorso()
    {
        return "non lo so";
    }
    /// <summary>
    /// Tratti di strada da evitare
    /// </summary>
    /// <param name="evita">punto da evitare</param>
    public void InserireDeviazione(string evita)
    {

    }
    /// <summary>
    /// Quanti Km a destinazione
    /// </summary>
    /// <returns>km</returns>
    public int DistanzaTotale()
    {
        return 55;
    }
    /// <summary>
    /// Tempo totale a destinazione
    /// </summary>
    /// <returns>ore</returns>
    public double TempoTotale()
    {
        return 1.5;
    }
    #endregion
}
```





Una soluzione migliore: gli oggetti

Mike ha scritto un nuovo programma per confrontare tre tragitti proposti da tre navigatori differenti per trovare il percorso più breve:

Mike ha creato un'applicazione Console dove vengono inseriti i vari valori in modo statico.

Navigatore
+Marca : string +Dimensioni : int +Modello : string
+InserireDestinazione() +TrovarePercorso() : string +InserireDeviazione() +DistanzaTotale() : int +TempoTotale() : double +InserirePOI()

```
string arrivo;
string percorso;
int km;
Navigatore tomTom;

tomTom = new Navigatore();

arrivo = "Milano";

tomTom.InserireDestinazione(arrivo);
percorso = tomTom.TrovarePercorso();
```

Compiti

1. Completa il progetto 01 Navigatore con i seguenti punti:
   - Aggiungere un secondo navigatore garmin. Chiamare il metodo InserireDestinazione e InserireDeviazione.
   - Aggiungere un terzo navigatore medion. Chiamare il metodo InserireDestinazione.
   - Definire un nuovo metodo InserirePOI (Point Of Interest), dove viene indicato il punto da cui si vuole passare obbligatoriamente e utilizzarlo con il navigatore medion.
   - ~~(Ora Mike chiama il metodo DistanzaTotale per i tre navigatori per scoprire il tragitto più breve.)~~

2. Crea un progetto con la classe **Clown** dove il metodo Descriviti prepara il nome del clown e la sua altezza. Stampa la descrizione dei seguenti 4 clowns (Dimitri; Buffo, 1.63; Baffo, 1.71; Beffo, 1.70) ~~[Setter – Getter]~~