

Progetto: Battaglia Navale

Classi:

```
C# Campo.cs
Form1.cs
C# Giocatore.cs
C# Nave.cs
C# Partita.cs
```

Classe Partita:

```
internal class Partita
{
    private const int dimensione = 6;
    public int nNavi = 0;

    public Campo campo = new Campo(dimensione, Campo.PlayerType.Io);
    public Campo campoAvversario = new Campo(dimensione, Campo.PlayerType.Nemico);

    private Giocatore g1;
    private Giocatore g2;

    public Giocatore[] giocatori = new Giocatore[2];

    1 riferimento
    public Partita(Giocatore g1, Giocatore g2)
    {
        this.g1 = g1;
        this.g2 = g2;

        giocatori[0] = g1;
        giocatori[1] = g2;

        g1.creaNaviPosizionabili(dimensione);
        g2.creaNaviPosizionabili(dimensione);
        nNavi = g1.navi.Count;
    }
}
```

- **Costruttore:** Partita(Giocatore g1, Giocatore g2);
- **Attributi:**
 - Campo campo: campo del giocatore umano,
 - Campo campoAvversario: campo del computer;
 - Giocatore[] giocatori: vettore contenente i due giocatori.

Classe Campo:

```
public class Campo
{
    Più di 99 riferimenti
    public enum CellType[...]

    Più di 99 riferimenti
    public enum PlayerType[...]

    public PlayerType PT;

    public CellType[,] campoIDs;

    public readonly int dimensione;
    20 riferimenti
    public Campo(int dimensione, PlayerType PT) [...]

    1 riferimento
    public void creaCampoIDs() [...]

    3 riferimenti
    public bool ciSta(Nave n, int x, int y) [...]

    3 riferimenti
    public bool InserisciNave(Nave naveSelezionata, int x, int y) [...]

    2 riferimenti
    public void AssegnaColpo(int i, int j) [...]
}
```

- **Costruttore:** Campo(int dimensione, PlayerType PT), crea il campo in base alla dimensione desiderata e in base al tipo di giocatore a cui assegnare il campo.

- Enum PlayerType:

```
public enum PlayerType
{
    Io, Nemico
}
```

- **Attributi:**

- Enum CellType: tipo di cella nel campo

```
public enum CellType
{
    Mare, Buco, Colpito, Affondato, Barca
}
```

- Enum PlayerType: tipi di player
- PlayerType PT: tipo di campo in base al tipo di player
- CellType[,] campoIDs: campo di id.
- Int dimensione: dimensione del campo

- **Metodi:**

- creaCampiIDs(): crea il campo inizializzato con inizialmente solo mare
- Bool ciSta(Nave n, int x, int y): ritorna se una nave ci sta in una posizione x, y;
- Bool InserisciNave(Nave n, int x, int y): inserisce la nave e ritorna se l'inserimento è riuscito;
- AssegnaColpo(int i, int j): cambia l'id della cella in quelle coordinate

Classe **Giocatore**:

```
class Giocatore
{
    public string nome;
    private Nave nave;
    public List<Nave> navi = new List<Nave>();

    2 riferimenti
    public Giocatore(string nome) [...]
    1 riferimento
    public void InserisciNavi(Campo c) [...]

    2 riferimenti
    public void creaNaviPosizionabili(int dimensione) [...]
}
```

- **Costruttore:** **Giocatore**(string nome): inserisci nome
- **Attributi:**
 - String nome: nome del giocatore;
 - List<Nave> navi: navi del giocatore
- **Metodi:**
 - **InserisciNavi**(Campo c): inserimento navi casuale;
 - **creaNaviPosizionabili**(int dimensione): crea le navi che possono essere inserite da un giocatore

Classe Nave:

```
public class Nave
{
    public int dimensione;

    private CellType[] nave;

    3 riferimenti
    private enum CellType[...]

    47 riferimenti
    public enum Orientamento[...]

    42 riferimenti
    public enum Type[...]

    public Type type;

    public Orientamento orientamento;

    20 riferimenti
    public Nave(Type type, Orientamento orientamento) [...]

    0 riferimenti
    public void setOrientamento(Orientamento o) [...]

    1 riferimento
    public void toggleOrientamento() [...]
```

- **Costruttore:** Nave(Type type, Orientamento orientamento): la nave ha un tipo (la lunghezza) e un orientamento (verticale od orizzontale)

```
public enum Type
{
    ...
    da2, da3, da4, da5
}
```

```
public enum Orientamento
{
    ...
    Verticale, Orizzontale
}
```

- **Attributi:**
 - Int dimensione: dimensione della nave
 - CellType[] nave: la nave è un vettore di tipo di cella, che può essere integra, colpita o affondata
- ```
private enum CellType
{
 ...
 Colpito, Affondato, Integra
}
```
- Enum Orientamento: la nave può essere orizzontale o verticale
- **Metodi:**
    - setOrientamento(Orientamento o): setta l'orientamento
    - toggleOrientamento(): toglia l'orientamento