Esercizio 18 - parte 1

Gli oggetti di classe **Pesce** sono caratterizzati dai seguenti attributi e metodi:

Attributi:

- nome:(string)
- peso:(double)
- colore:(string)
- velocità:(int): rappresenta una misura della distanza percorsa con un colpo di pinna posteriore
- habitat (boolean): vale true se e solo se l'esemplare è un pesce di mare, false se è di fiume o lago
- stile (char): che rappresenta lo stile di nuotata

Metodi:

- costruttore con 0 parametri
- costruttore con 5 argomenti che inizializza l'oggetto con valori dall'esterno tutti gli attributi eccetto l'attributo stile, al quale assegna il carattere '.';
- toString(): che restituisce una stringa con i dati del pesce.
- isMare(): che restituisce true se e solo se il pesce vive in acqua salata;
- isDolce(): che restituisce true se e solo se il pesce è d'acqua dolce;
- getNome(): che restituisce il nome dell'esemplare;
- getPeso(): che restituisce il peso dell'esemplare;
- getColore(): che restituisce il colore dell'esemplare.
- getVelocità(): che restituisce la velocità dell'esemplare.
- getStile(): che restituisce lo stile dell'esemplare.
- getNuotata(): che fornisce una rappresentazione grafica in ASCII art dello stile di nuotata del pesce. Il metodo restituisce una stringa costituita da n ripetizioni del carattere "stile", dove n = 1+(10*vel/INT((peso+2)/2)), dove INT(x)è la parte intera di x. Ad esempio, se lo stile di nuotata è ')', la stringa restituita potrebbe avere la forma ")))))))))";

Derivare dalla classe Pesce le classi Salmone, PescePalla e PesceGatto basandosi sui dati seguenti

Il salmone è un pesce d'acqua dolce, lo stile di nuotata è identificato dal carattere '/' e possiede un attributo aggiuntivo di nome "fiume" (string), che ne rappresenta il fiume d'origine;
Ha i seguenti metodi pubblici:

- costruttore con 4 argomenti che inizializza l'oggetto con il peso, il colore, la velocità e il fiume d'origine (impostando gli altri campi ai valori opportuni, es. nome = "salmone");
- getFiume(): che restituisce il valore del campo fiume.
- ToString(): che restituisce una stringa con i dati del pesce.

Il PescePalla è un pesce di mare, lo stile di nuotata è identificato dal carattere ')' e possiede un attributo aggiuntivo di nome di nome "spaventato" (boolean), inizialmente settato a false;
Ha i seguenti metodi pubblici:

- costruttore con 3 argomenti che inizializza l'oggetto con il peso, il colore e la velocità (impostando gli altri campi ai valori opportuni in modo analogo al costruttore di Salmone);
- isSpaventato(): che restituisce il valore del campo spaventato.
- Spaventa(): che assegna true al campo spaventato.
- Tranquillizza(): che assegna false al campo spaventato.
- ToString(): che restituisce una stringa con i dati del pesce.

- GetNuotata(): restituisca la stringa corrispondente alla nuotata, ma restituisce la stringa vuota quando l'esemplare è spaventato (poiché si gonfia e non è più in grado di nuotare).

Il PesceGatto è un pesce d'acqua dolce e lo stile di nuotata è identificato dal carattere '>';
Ha i seguenti metodi pubblici:

- costruttore a 3 argomenti che inizializza l'oggetto con il peso, il colore e la velocità (impostando gli altri campi ai valori opportuni in modo analogo al costruttore di Salmone);
- getNuotata(): restituisce stringhe di lunghezza doppia rispetto a quello della classe Pesce, perché è più agile degli altri.
- ToString(): che restituisce una stringa con i dati del pesce.

Nota: eventualmente sostituire i metodi getter e setter con properties.

Realizzare le classi Pesce, Salmone, PescePalla e PesceGatto, secondo le specifiche sopra riportate, e un'applicazione console che funga da codice consumer per testare la classe.
Variante: Realizzare una applicazione consumer con GUI.

Esercizio 18 - parte 2

Modificare, se necessario, il codice delle classi dell'esercizio 1 applicando i principi del polimorfismo e creare una classe Acquario (contenitore di oggetti pesce) avente le seguenti caratteristiche:

Attributi:

- capienza:(double) capienza massima in litri
- tipologia:(string) acqua salata, acqua dolce, ecc ecc
- pesciMax: (int) numero massimo di pesci ospitabili, dato dalla formula *capienza * 4*;
- pesci: (array di oggetti pesce)

Metodi:

- costruttore con 0 parametri
- costruttore con 2 parametri, capienza e tipologia
- AggiungiPesce(Pesce): aggiunge un pesce all'acquario, ritorna un bool
- AggiungiPesci(collezione di oggetti Pesce): aggiunge una collezione di pesci all'acquario, ritorna un bool
- AggiungiPesceCasuale(): nessun parametro, aggiunge in maniera casuale un pesce tra le tipologie disponibili, ritorna una bool
- SpaventaPesci(): void, spaventa tutti i pesci palla presenti in acquario
- PesoTotale(): restituisce la somma del peso dei pesci presenti
- ToString(): override dell'omonimo metodo che ritorna le caratteristiche dell'acquario

Realizzare un'applicazione console che funga da codice consumer per testare la classe.

Per verificare di aver applicato correttamente i principi del polimorfismo provare ad invocare il metodo toString() e getNuotata() su tutti i pesci presenti nell'acquario.

NOTE:
Aggiungere eventuali properties per migliorare l'usabilità della classe.
I metodi per aggiungere i pesci devono aggiungere nuovi pesci soltanto fino al raggiungimento del massimo numero di pesci consentito.
Tralasciare il fatto che nello stesso acquario siano presenti pesci d'acqua dolce e di acqua salata e comunque pesci che non sono notoriamente pesci da acquario.

Variante: Realizzare una applicazione consumer con GUI.