Object kata

Indice generale

1 Punto in coordinate cartesiane (immutabile)

Problema

Implementa un *oggetto* immutabile che rappresenti un punto in coordinate cartesiane. L'*oggetto* deve fornire un metodo di traslazione che accetti la distanza, in X e in Y, dello spostamento. Il metodo deve restituire un nuovo punto che equivalga al vecchio dopo lo spostamento. Esempio:

```
Punto p = new Punto(10, 50);
// p.X->10   p.Y->50

Punto p2 = p.Trasla(-5, 20);   // "p" non cambia coordinate
Console.Write("({0}, {1})", p2.X, p2.Y);
// ->(5,70)
```

Variazioni

Aggiungere un metodo, `ToString()`, che produca la rappresentazione stringa del punto: `x;y`.

Aggiungere un metodo statico, `Parse()`, che crei un punto sulla base di una stringa nel formato: `x;y`.

Aggiungere gli operatori == e !=, i quali consentono di confrontare due oggetti.

2 Oggetto Rettangolo

Problema

Implementa un *oggetto* mutabile che rappresenti un rettangolo, caratterizzato da:

- le coordinate x,y del vertice in alto a sinistra.
- Le dimensioni: larghezza e altezza.

L'*oggetto* deve definire delle proprietà scrivibili che restituiscano i valori suddetti, garantendo l'invariante di classe sulla base delle seguenti condizioni:

- x >= 0; y >= 0
- larghezza > 0; altezza > 0

Deve inoltre definire delle proprietà derivate che restituiscano le coordinate massime, *destra* e *basso*, del rettangolo.

Esempio:

```
Rettangolo  r = new Rettangolo(10, 20, 100, 50);
// r.X->10  r.Y->20  r.Larghezza->100   r.Altezza->50  r.Destra = 110  r.Basso->70
```

Variazioni

Aggiungere il metodo `Inflate(int width, int height)`, che ridimensiona il rettangolo in base ai valori specificati (il metodo deve garantire l'invariante di classe). Esempio:

```
Rettangolo  r = new Rettangolo(10, 20, 100, 50);
r.Inflate(5, 10);  // aumenta la dimensione in X di 10 e in Y di 20
// r.X->15  r.Y->30  r.Larghezza->110   r.Altezza->70
```

Aggiungere il metodo `Contains(Rettangle r)`, che restituisce `true` se il rettangolo contiene completamente il parametro specificato. Ad esempio:

```
Rettangolo  r = new Rettangolo(10, 20, 100, 50);
Rettangolo  r2 = new Rettangolo(12, 21, 50, 30);  // è contenuto nell'area di "r"
Rettangolo  r3 = new Rettangolo(12, 21, 99, 30);  // NON è contenuto nell'area di "r"
bool contiene = r.Contains(r2);  // -> true
contiene = r.Contains(r3);  // -> false
```

3 Classe Email

Problema

Implementa un *oggetto* immutabile in grado di incapsulare un indirizzo e-mail. L'*oggetto* deve fornire delle proprietà che restituiscano:

- l'indirizzo completo.
- Nome utente.
- dominio.

La classe deve implementare un costruttore che accetta una stringa contenente l'indirizzo. Il costruttore dovrà validare l'argomento, che deve rispettare il seguente formato:

`<nome-utente>@<dominio>`

Alcuni esempi: `filippo.rossi@gmail.com`, `superman@libero.it`, `thenerdy76@virgilio.it`

Esempio:

```
Email mail = new Mail("rosa.bianchi89@hotmail.it");
// -> nome utente: rosa.bianchi     dominio: hotmail.it

Console.Write("[{mail.Utente}] [{mail.Dominio}] [{mail.Address}]");
// -> [rosa.bianchi89] [hotmail.it] [rosa.bianchi89@hotmail.it]

mail = new Mail("rosa.bianchi89@");
// -> FormatException()
```

Variazioni

Aggiungere un metodo, `ToString()`, che produca l'indirizzo completo.

Aggiungere gli operatori == e !=, i quali consentono di confrontare due oggetti. Nota bene: nel confrontare due indirizzi non si deve fare distinzione tra maiuscole e minuscole.

4 Classe PersistedDictionary

Problema

Implementa un dizionario in grado di persistere su disco le informazioni. Il dizionario dovrà:

- Gestire in memoria le informazioni mediante un oggetto `Dictionary<string, string>`, in modo da velocizzare l'accesso alle voci.
- Rimuovere immediatamente dal file ogni voce che viene cancellata.
- Salvare immediatamente su file ogni voce che viene aggiunta.

La classe deve definire un costruttore che accetta il nome del file, e i metodi: `Add()`, `Remove()`, la proprietà `Count` e un metodo per accedere a un valore data la sua chiave:

Esempio:

```
var dic = new PersistedDictionary("Config.ini");
dic.Add("forecolor","yellow");
dic.Add("backcolor","blue");

var dic1 = new PersistedDictionary("Config.ini");
var fr = dic1.GetValue("forecolor");
// -> "yellow"
var bk = dic1.GetValue ("backkcolor");   // chiave inesistente
// -> null
bool rimossa = dic1.Remove("forecolor");

dic1 = new PersistedDictionary("Config.ini");
fr = dic1.GetValue ("forecolor");
// -> null
```

Variazioni

Aggiungere un indicizzatore per accedere al valore data una chiave. Diversamente dal metodo `GetValue()`, questo dovrà sollevare un eccezione se la chiave non esiste.,