FACE ID

Hint: You can use GetHashCode(), Equals(), HashSet()

Instructions

You are working on a system to simplify the login process for your organization's network. The tasks concern the authentication part. The system uses facial recognition to prove identity.

In all occurrences the eye color parameter is guaranteed to be non-null.

1. Ensure that facial features match

Implement `Authenticator.AreSameFace()` to check that two faces match.

Add equality routines for the `FacialFeatures` class.

```
Authenticator.AreSameFace(new FacialFeatures("green",
0.9m), new FacialFeatures("green", 0.9m);
// => true
Authenticator.AreSameFace(new FacialFeatures("blue",
0.9m), new FacialFeatures("green", 0.9m);
// => false
```

2. Authenticate the system administrator

Despite your protests the system administrator insists on having a built-in identity during acceptance testing so that they can always be authenticated.

The admin's email is admin@exerc.ism. They have green eyes and a philtrum with a width of 0.9.

Add equality routines for the `Identity` class.

Implement the `Authenticator.IsAdmin()` method to check that the identity passed in matches that of the administrator.

```
var authenticator = new Authenticator();
authenticator.IsAdmin(new Identity("admin@exerc.ism", new
FacialFeatures("green", 0.9m)));
// => true
authenticator.IsAdmin(new
Identity("admin@thecompetition.com", new
FacialFeatures("green", 0.9m)));
// => false
```

3. Register new identities

Implement the `Authenticator.Register()` method which stores an identity on the authenticator itself such that calls to `IsRegistered()` will return `true` for this identity: otherwise `IsRegistered()` returns `false`.

To detect duplicated attempts to register an identity, if the identity has already been registered then `false` is returned by `Authenticator.Register()`, otherwise `true`.

```
var authenticator = new Authenticator();
authenticator.Register(new
Identity("tunde@thecompetition.com", new
FacialFeatures("blue", 0.9m)));
// => true
```

```
authenticator.IsRegistered(new
Identity("tunde@thecompetition.com", new
FacialFeatures("blue", 0.9m)));
// => true
authenticator.Register(new
Identity("tunde@thecompetition.com", new
FacialFeatures("blue", 0.9m)));
// => false
```

4. Prevent invalid identities being authenticated

Implement the `Authenticator.IsRegistered()` method and ensure it returns false when no identities have been registered.

```
var authenticator = new Authenticator();
authenticator.IsRegistered(new
Identity("alice@thecompetition.com", new
FacialFeatures("blue", 0.8m)));
// => false
```

5. Add diagnostics to detect multiple attempts to authenticate

A bug has been reported whereby the `Authenticator.IsRegistered()` method is called multiple times in quick succession for the same identity. You believe that there is some sort of "bounce" problem where the exact same record is being submitted multiple times. Your task is to add a diagnostic routine `Authenticator.AreSameObject()` to support any testing that's undertaken. The routine compares to objects and returns `true` if they are the exact same instance otherwise `false`.

```
var identityA = new Identity("alice@thecompetition.com",
new FacialFeatures("blue", 0.9m));
var identityB = identityA;
Authenticator.AreSameObject(identityA, identityB);
// => true

var identityC = new Identity("alice@thecompetition.com",
new FacialFeatures("blue", 0.9m));
var identityD = new Identity("alice@thecompetition.com",
new FacialFeatures("blue", 0.9m));
Authenticator.AreSameObject(identityC, identityD);
// => false
```