名前の衝突

インターフェイスの問題点

インターフェイスには、ある特定の状態になると問題が発生します
それは、基底クラスとインターフェイスで同一のシグネチャが存在する場合
それぞれを継承、実装したクラスで名前の衝突が発生するというものです

```
interface A {
 string Name { get; }
}

class B {
 private const string name = "Kitty on your lap";
 public string Name { get { return name; } }
}

class C : B , A {}

class Test {
 static void Main() {
  C obj = new C();
  System.Console.WriteLine(obj.Name);
 }
}
```

クラス C は、インターフェイス A を実装しなければなりませんが
クラス C は A で定義されているプロパティ Name を実装していません
ところが、このプログラムは正常にコンパイルされてしまいます

これは、コンパイラは B クラスの Name プロパティを A の実装と勘違いしてしまうからです
だからといって、B クラスの設計者はインターフェイス A の仕様にしたがって
Name プロパティを作ったという保証はどこにもありません
なぜならば、B クラスは A インターフェイスを実装していないからです
B.Name プロパティは A インターフェイスとはまったく関連がないと考えることができます

開発者は、インターフェイスとクラスの関係にこのような問題があることを認識しなければなりません
そこで、オブジェクトは常に**目的の型にキャスト**して用いる癖をつけることです
それだけでも、継承関係で競合した名前などでは適切なメソッドを呼び出せます

```
interface A {
 void Write();
}

interface B {
 void Write();
}

class Kitty : A , B {
 void A.Write() {
  System.Console.WriteLine("Kitty on your lap");
 }
 void B.Write() {
  System.Console.WriteLine("Di Gi Charat");
 }
}

class Test {
 static void Main() {
  System.Object obj = new Kitty();
  ((A)obj).Write();
  ((B)obj).Write();
 }
}
```

このプログラムでは、インターフェイス A と B で同名のメソッドを定義しています
この二つのインターフェイスを同時に一つのクラスで実装すると名前が衝突するでしょう
しかし、Kitty クラスはクラスからインターフェイスの実装を隠蔽することで
それぞれの実装を区別し、適切なメソッドを呼び出せるように工夫してあります

となれば、あとはオブジェクトを適切なインターフェイス型にキャストすることで
それぞれのインターフェイスの適切なメソッド（実装）を呼び出すことができます

ただし、次のような場合はコンパイルすることすらできません
同じシグネチャを持つ二つのインターフェイスを結合したインターフェイス型の場合
どちらのメソッドを呼び出してよいのか判断できず、曖昧になってしまうからです

```
interface A {
 void Write();
}

interface B {
 void Write();
}

interface C : A , B {}

class Kitty : C {
 void A.Write() {
  System.Console.WriteLine("Kitty on your lap");
 }
```

```
 void B.Write() {
  System.Console.WriteLine("Di Gi Charat");
 }
}

class Test {
 static void Main() {
  C obj = new Kitty();
  obj.Write();
 }
}
```

C 型の参照は A と B 型の Write() メソッドの実装を持ちます
しかし、C 型のオブジェクトから Write() を呼び出すと
どの Write() メソッドを呼び出してよいか判断できず、コンパイルすることができないのです

前のページへ　戻る　次のページへ