บทที่ 12
Interface Class

ในการใช้งานเชิงวัตถุ จากบทเรียนที่ผ่านมา จะเห็นได้ว่าคลาสประกอบไปด้วย คลาสหลัก และคลาสย่อย ซึ่งในการการสร้างคลาสย่อยนั้นจะต้องต้องใช้คำสั่ง Extend จากจากคลาสหลักได้เพียงคลาสเดียวเท่านั้น ซึ่งคลาสย่อยจะได้รับเมธอด และแอตทริบิวต์ จากคลาสหลักมาด้วย ซึ่งเรียกว่า การสืบทอดคลาส

การเขียนโปรแกรมแบบเชิงวัตถุ ยังมีลักษณะการใช้งานที่เป็นลักษณะที่คล้ายกับการสืบทอดคลาสที่กล่าวมา แต่ในการสืบทอดคลาสนั้น คลาสลูกสามารถสืบทอดจากคลาสหลักได้หลายคลาสในครั้งเดียวกัน แต่จะสามารถสืบทอดคลาสได้เพียงในส่วนของเมธอดเท่านั้น

12.1 อินเทอร์เฟสคลาสเดียว

การอินเทอร์เฟสคลาสเดียว หมายถึง การสร้างคลาสย่อยจากคลาสหลัก คลาสเดียว ซึ่งมีลักษณะของการทำงาน ดังตัวอย่างตารางที่ 12.1

ตารางที่ 12.1 การสร้างคลาสประเภทอินเทอร์เฟซคลาส

```
1   <?
2   interface land_animal
3   {
4           public function leg();
5           public function lung();
6           public function cold_blooded();
7   }
8
9   class elephant implements land_animal
10  {
11          public function leg()
12          {
13                  echo "เมธอด leg ที่ได้จาก อินเทอร์เฟสคลาส  land_animalmom";
14          }
```

```
15          public function lung()
16          {
17          }
18          public function cold_blooded()
19          {
20          }
21  }
22  $elephant1 = new elephant();
23  $elephant1->leg();
24  ?>
```

ผลลัพธ์

```
เมธอด leg ที่ได้จาก อินเทอร์เฟสคลาส land_animalmom
```

จากตัวอย่างตารางที่ 12.1 แสดงลักษณะของการใช้งานแบบอินเทอร์เฟสคลาส ซึ่งในการสร้างคลาสแบบนี้ จะต้องใช้คำสั่ง interface นำหน้าชื่อคลาสที่สร้าง (บรรทัดที่ 2) ซึ่งในอินเทอร์เฟสคลาสจะประกอบไปด้วยชื่อแอททริบิวต์ จากตัวอย่างเมื่อมีการสร้างคลาส elephant ที่เกิดขึ้นมา อินเทอร์เฟสคลาส land_animal จะต้องใช้คำสั่ง implements (ซึ่งจะแตกต่างจาก extend ที่เคยกล่าวมาในเบื้องต้น) ตามด้วยชื่อของอินเทอร์เฟสคลาส land_animal และในคลาส elephant จะมีเมธอดเป็นชื่อเดียวกันที่มีอยู่ในอินเทอร์เฟสคลาส land_animal ด้วยนั่นคือ leg, lung และ cold_blooded เมื่อมีการสร้างออบเจ็กต์ขึ้นมา และมีการเรียกใช้เมธอด ก็จะได้ผลลัพธ์ดังตัวอย่าง

ตัวอย่างที่ 12.2 การสร้างเมธอดในคลายย่อยไม่ครบตามเมธอดอินเทอร์เฟซคลาส

```
1   <?
2   interface land_animal
3   {
4           public function leg();
5           public function lung();
6           public function cold_blooded();
7   }
8
9   class elephant implements land_animal
10  {
11          public function leg()
```

```
12          {
13                  echo "เมธอด leg ที่ได้จาก อินเทอร์เฟสคลาส  land_animalmom";
14          }
15  }
16  $elephant1 = new elephant();
17  $elephant1->leg();
18  ?>
```

ผลลัพธ์ของโปรแกรม

Fatal error: Class elephant contains 2 abstract methods and must therefore be declared abstract or implement the remaining methods (land_animal::lung, land_animal::cold_blooded) in **C:\AppServ\www\webapp\interface_class.php** on line **15**

ในส่วนของตัวอย่างในตารางที่ 2 จะสังเกตได้ว่า ในอินเทอร์เฟสคลาส mom จะประกอบไปด้วยเมธอด 3 เมธอด แต่เมื่อมีการสร้างคลาส child ที่เกิดจากอินเทอร์เฟสคลาส mom และเมื่อนำไปสร้างเมธอด ในคลาสของตนเอง มีการนำเอาชื่อเมธอด name ไปสร้างเพียงเมธอดเดียว ผลลัพธ์ของโปรแกรมจะแสดงผิดพลาดขึ้นมาดังตัวอย่าง จากผลลัพธ์ตรงนี้หมายถึงว่า เมื่อมีการสร้างคลาสจากอินเทอร์เฟสคลาสมาแล้ว เมื่อมีการสร้างเมธอดในคลาส จะต้องมีจำนวนเมธอด และชื่อของเมธอดทั้งหมดตามอินเทอร์เฟสคลาสมีด้วย

12.2 การอินเทอร์เฟสมากกว่าหนึ่งคลาส

ประโยชน์ที่สำคัญของอินเทอร์เฟซคลาสนั่นก็คือ ปกติคลายย่อยหนึ่งคลาสจะเกิดมาจากคลาสหลักเพียงคลาสเดียวเท่านั้น แต่ในการใช้งานของลักษณะอินเทอร์เฟซคลาส คลายย่อยหนึ่งคลาสสามารถที่จะเกิดจากคลาสหลักได้หลายคลาส ดังตัวอย่างตารางที่ 12.3

ตารางที่ 12.3

```
1   <?
2   interface land_animal
3   {
4           public function leg();
5           public function lung();
6           public function cold_blooded();
7   }
```

```
8  interface aquatic_animal
9  {
10         public function live_water();
11         public function tail();
12 }
13 interface wing_animal
14 {
15         public function wing();
16         public function tip();
17 }
18 class penguin implements land_animal,aquatic_animal,wing_animal
19 {
20         public function leg() {
21                 echo "มี 2 ขา";
22         }
23         public function lung() {
24                 echo "มีปอด";
25         }
26         public function cold_blooded()        {
27                 echo "เป็นสัตว์เลือดเย็น";
28         }
29         public function live_water(){
30                 echo "อาศัยในน้ำ";
31         }
32         public function tail(){
33                 echo "มีหาง";
34         }
35         public function wing(){
36                 echo "มีปีก";
37         }
38         public function tip(){
39                 echo "มีจะงอย";
40         }
```

```
41  }
42  $penguin1 = new penguin();
43  $penguin1->leg(); echo "<br>";
44  $penguin1->lung(); echo "<br>";
45  $penguin1->cold_blooded(); echo "<br>";
46  $penguin1->live_water(); echo "<br>";
47  $penguin1->tail(); echo "<br>";
48  $penguin1->wing(); echo "<br>";
49  $penguin1->tip(); echo "<br>";
50  ?>
```

จากตัวอย่างตารางที่ 12.3 จะเห็นได้ว่าบรรทัดที่ 12 – 17 ประกอบไปด้วยอินเทอร์เฟซคลาสทั้งหมด 3 อินเทอร์เฟซคลาส ประกอบไปด้วย land_animal aquatic_animal และ wing_animal บรรทัดที่ 18 เป็นการสร้างคลาสย่อยเกิดจากอินเทอร์เฟซคลาสทั้งสาม ในการอ้างอิงอินเทอร์เฟซคลาสจะใช้สัญลักษณ์ “,” (คอมมา) ในการคั่นอินเทอร์เฟซคลาสในแต่ละตัว และเมธอดจะมีจำนวนตามเมธอดที่อยู่ในแต่ละอินเทอร์เฟซคลาสตามจำนวนในแต่ละตัว ดังบรรทัดที่ 20-40