Lab 5 Inheritance and Polymorphism

ข้อ 1 คลาส Main

```
1 package oot.lab5;

2 public class Main {
3    public static void main(String[] args) {
4        Member manee = new Member("Manee", new Account("A111", 0));
5        Member mana  = new Member("Mana",  new Account("A123", 0), "Normal");
6        Account maneeAccount = manee.getAccount();
7        Account manaAccount  = mana.getAccount();
8        maneeAccount.buy(5);
9        manaAccount.buy(10);
10       System.out.println(manee.getName() + " has id: " + maneeAccount.getId() + " (Status:" +
manee.getStatus() + ")");
11       System.out.println(manee.getName() + " buy " + maneeAccount.getTotal() + " items.");
12       System.out.println();
13       System.out.println(mana.getName() + " has id: " + manaAccount.getId() + " (Status:" +
mana.getStatus() + ")");
14       System.out.println(mana.getName() + " buy " + manaAccount.getTotal() + " items.");
15   }
16}
```

บรรทัดที่ 1 ประกาศแพ็กเกจ (package) ในที่นี้คือ oot.lab5 โดยแพ็กเกจคือโครงสร้างเป็นระดับชั้นคล้ายโฟลเดอร์ (หรือไดเร็กทอรี) ที่ใช้จัดเก็บไฟล์ ในภาษาจาวา แพ็กเกจใช้จัดเก็บหมวดหมู่ของคลาส (class)

บรรทัดที่ 2 เป็นการ ประกาศ คลาส ชื่อ Main ให้เป็นคลาสประเภท public (สาธารณะ) คือสามารถใช้ได้ทั่วไป หมายความว่าคลาสอื่น ๆ สามารถอ้างถึงคลาสประเภทนี้เพื่อใช้งานคลาสประเภทนี้ได้

บรรทัดที่ 3 เป็นการประกาศเมธอดชื่อ main เป็นชนิด void และเป็นประเภท static โดยรับพารามิเตอร์ชื่อ args เป็นชนิด อะเรย์ของ String ซึ่งเป็นข้อกำหนด ในภาษา Java ว่าการประกาศเมธอดแบบนี้เท่านั้นจึงจะใช้เป็นจุดเริ่มต้นของโปรแกรมได้

บรรทัดที่ 4 เป็นการสร้างวัตถุที่มีชนิด (type) เป็นคลาส Member แล้วเก็บค่าวัตถุนี้ไว้ในตัวแปรชื่อ manee เรารู้ว่าเป็นการสร้างวัตถุ
เนื่องจากทางขวามือของบรรทัดนี้เป็นการใช้ new

- การสร้างวัตถุจะต้อง (1) เขียน new ตามด้วย (2) คอนสตรักเตอร์ของคลาสที่ต้องการจะสร้าง
- อาจจะมีตัวแปรมารับวัตถุนั้นก็ได้ เช่น manee เป็นตัวแปรมารับวัตถุที่สร้างในบรรทัดที่ 4
- ถ้ามีตัวแปรมารับ ตัวแปรต้องเป็นชนิดเดียวกันกับคลาสที่ใช้สร้างวัตถุ (ตัวแปรอยู่ซ้ายมือ และมีเครื่องหมายเท่ากับ และมี new ตามด้วยคอนสตรักเตอร์ทางขวามือ)

```
Member manee = new Member("Manee", new Account("A111", 0));
```

- ในบรรทัดนี้ ใช้คอนสตรักเตอร์ของคลาส Member ที่รับพารามิเตอร์สองตัว
    1. ตัวแรกมีชนิดเป็น String คือคำว่า “Manee”
    2. ตัวที่สองมีชนิดเป็น Account คือวัตถุ new Account(“A111”, 0) - เรารู้ว่าวัตถุนี้เป็นวัตถุของคลาส Account เพราะใช้ new แล้วตามด้วยคอนสตรักเตอร์ชื่อ Account(...) นั่นเอง

ด้วยข้อมูลจากบรรทัดที่ 4 ทำให้เรารู้ว่า

1. จะต้องมีการประกาศคลาสชื่อ Member ไว้ในแพ็กเกจเดียวกันกับคลาส Main - เพราะไม่มีการใช้ import เพื่ออ้างถึงคลาส Member แปลว่าคลาสทั้งสองอยู่ในแพ็กเกจเดียวกัน (ดูบรรทัดที่ 1 และ 2 ของคลาส Member)
2. คลาส Member จะต้องมีการประกาศคอนสตรักเตอร์อย่างน้อยหนึ่งตัวไว้ในคลาสนั้น และคอนสตรักเตอร์นั้นต้องรับตัวแปรชนิด String เป็นพารามิเตอร์ที่ 1 และ ตัวแปรชนิด Account เป็นพารามิเตอร์ที่ 2 (ดูบรรทัดที่ 6 ของคลาส Member)

บรรทัดที่ 5 เป็นการสร้างวัตถุที่มีชนิดเป็นคลาส Member แล้วเก็บค่าวัตถุนี้ไว้ในตัวแปรชื่อ mana คล้ายกับการทำงานในบรรทัดที่ 4
จุดที่แตกต่างระหว่างโปรแกรมบรรทัดที่ 4 กับ บรรทัดที่ 5 คือ การใช้คอนสตรักเตอร์คนละตัวกัน โดยในบรรทัดนี้จะมีการใช้คอนสตรักเตอร์ของคลาส Member ที่รับพารามิเตอร์ได้ 3 ตัว

```
Member mana  = new Member("Mana",  new Account("A123", 0), "Normal");
```

เราสามารถวิเคราะห์ได้ว่าคอนสตรักเตอร์นี้รับพารามิเตอร์ตัวแรกเป็นชนิด String เพราะค่า “Mana”
สำหรับพารามิเตอร์ตัวที่สองจะเป็นชนิด Account เพราะวัตถุ new Account(“A123”, 0)
และพารามิเตอร์ตัวที่สามจะเป็นชนิด String เพราะค่า “Normal”
ซึ่งคอนสตรักเตอร์ ตัวนี้ประกาศอยู่ที่บรรทัดที่ 12 ของคลาส Member

เทคนิค - วิธีอ่านเบื้องต้น

- Member เป็นคลาส
- new Member(...) เป็นวัตถุของคลาส Member ไม่ว่าจะถูกเขียนไว้ที่ใดก็ตาม
- Account เป็นคลาส
- new Account(…) เป็นวัตถุของคลาส Account ไม่ว่าจะถูกเขียนไว้ที่ใดก็ตาม
- Member manee = new Member(...) แบ่งได้เป็นสองส่วน
    - ซ้ายมือได้วัตถุของคลาส Member
    - แล้วนำวัตถุของคลาส Member ที่สร้างขึ้นให้ตัวแปร manee ทางขวามือ
    - ข้อสังเกต เหมือนเราเขียน Integer i = 0; แต่ในกรณีนี้เรามองไม่เห็นคอนสตรักเตอร์
    - ข้อสังเกต เหมือนเราเขียน String name = “Somsak”; แต่เราไม่เห็นคอนสตรักเตอร์

บรรทัดที่ 6 เป็นการใช้เมธอดประเภท getter ชื่อ getAccount() เพื่อดึงค่าฟิลด์ชื่อ account ที่อยู่ภายในตัวแปรวัตถุชื่อ manee ออกมา
getter เป็นเมธอดตามข้อตกลงในภาษา Java ที่กำหนดว่า

- ถ้าต้องการให้มีเมธอดสำหรับดึงค่าออกมาจากฟิลด์
    - เมธอดควรจะขึ้นต้นด้วยคำว่า get
    - แล้วตามด้วยชื่อฟิลด์ที่ต้องการดึง และเปลี่ยนตัวหน้าของชื่อฟิลด์ให้เป็นตัวใหญ่

นั่นคือ มีฟิลด์ชื่อ account ก็จะได้ getAccount() เป็นชื่อ getter

จากบรรทัดที่ 6 นี้ ทำให้เรารู้ว่า ตัวแปร manee ซึ่งเป็นวัตถุของคลาส Member น่าจะมีฟิลด์ชื่อ account และฟิลด์นี้มีชนิดเป็นคลาส Account - เพราะในบรรทัดนี้ต้องมีการประกาศตัวแปรชื่อ maneeAccount มารับค่าจากการเรียกใช้ getter ชื่อ getAccount() นั่นเอง

```
Account maneeAccount = manee.getAccount();
```

เทคนิค - วิธีอ่านและวิเคราะห์ชนิดตัวแปร

ถ้าเขียน manee.getAccount() แล้วเรารู้ว่า manee คือวัตถุคลาส Member มองง่าย ๆ คือ <วัตถุคลาส Member ชื่อ manee>.getAccount()
อ่านว่า “เรียกใช้ เมธอด getAccount() ของคลาส Member โดยให้ตัวแปร manee เป็นตัวแปร this ในเมธอด getAccount”
แปลว่าในคลาส Member ต้อง มีเมธอดชื่อ getAccount ที่ไม่มีพารามิเตอร์ (ดูบรรทัดที่ 33 ของคลาส Member)
และในเมธอด getAccount() มีการคืนค่า this.account โดยให้ตัวแปร manee เป็นตัวแปร this ชั่วคราวในช่วงเวลาการใช้เมธอดนี้
ทำให้เราจะได้ค่าฟิลด์ของวัตถุ manee ออกมา

และจากการที่ประโยคในบรรทัดที่ 6 มีเครื่องหมายเท่ากับ ทำให้ตัวแปร maneeAccount บรรจุค่าเดียวกันกับค่าของฟิลด์ account ที่เก็บอยู่ในวัตถุ manee
คำถามก็คือ ค่าอะไรเก็บอยู่ในฟิลด์ account?
คำตอบอยู่ที่บรรทัดที่ 4 ของคลาส Main ซึ่งเป็นบรรทัดที่สร้างวัตถุให้ตัวแปร manee
และเราต้องดูบรรทัดที่ 6, 9 ของคลาส Member ประกอบกัน
ทำให้เรารู้ว่าตัวแปร maneeAccount มีค่าเป็นวัตถุ new Account(“A111”, 0)

เทคนิค - มองทุกอย่างเป็นวัตถุ

เลข 0 คือ new Integer(0)

คำว่า “sut” คือ new String(“sut”)

บรรทัดที่ 7 อธิบายได้ด้วยวิธีการเดียวกันกับบรรทัดที่ 6
ดูโปรแกรมของคลาส Main บรรทัดที่ 5
ประกอบกับคลาส Member บรรทัดที่ 12, 15

```
Account manaAccount  = mana.getAccount();
```

ตัวแปรวัตถุ mana ถูกนำไปเป็นตัวแปร this ชั่วคราวในการเรียกใช้เมธอด getAccount() ของคลาส Member
ทำให้ได้ฟิลด์ account ของวัตถุ mana ออกมา ค่าดังกล่าวคือวัตถุ new Account("A123", 0)

บรรทัดที่ 8 เป็นการเรียกใช้เมธอด buy(int) ของคลาส Account บนตัวแปรวัตถุ maneeAccountโดยส่งค่า 5 ไปให้พารามิเตอร์ที่ 1 และค่าดังกล่าวต้องเป็น int

```
maneeAccount.buy(5);
```

เรารู้ได้อย่างไรเมธอด buy เป็นเมธอดของคลาส Account? เหตุผลคือตัวแปร maneeAccount ถูกประกาศเป็นชนิด Account ไว้ในบรรทัดที่ 6 ของคลาส Main
การเรียกใช้เมธอด buy(int) ในบรรทัดนี้สามารถอ่านได้ในทำนองเดียวกันกับการเรียกใช้เมธอด getAccount()
อ่านว่า "เรียกใช้เมธอด buy(int) ของคลาส Account โดยให้ตัวแปร maneeAccount เป็นตัวแปร this ตลอดการทำงานในเมธอด buy(int) และให้พารามิเตอร์ตัวที่ 1 มีค่าเป็น 5"

เมื่อเราดูโปรแกรมในคลาส Account บรรทัดที่ 10 ตรงการประกาศเมธอด buy(int) จะเห็นว่าพารามิเตอร์ตัวที่ 1 มีการตั้งชื่อว่า amount นั่นคือ ค่า 5 ในบรรทัดที่ 8 นี้จะไปอยู่ในตัวแปร amount

ในบรรทัดที่ 11 ของคลาส Account จะมีการอ้างถึง this.total แล้วนำค่า amount ไปบวกเพิ่ม ตรงจุดนี้จะอธิบายได้ว่าการเรียกใช้เมธอด buy ในบรรทัดที่ 8 นี้จะทำให้ตัวแปร this กลายเป็นตัวแปรเดียวกันกับ maneeAccount นั่นหมายความว่าฟิลด์ total (จากการเขียน this.total) จะเป็นฟิลด์ของวัตถุ maneeAccount เราจะไล่ย้อนกลับไปว่าฟิลด์ total นี้มีค่าเป็นเท่าใดจากการดูโปรแกรมบรรทัดที่ 6 ของคลาส Main

เมื่อย้อนกลับไปที่บรรทัดที่ 6 จะเห็นว่าตัวแปร maneeAccount เป็นวัตถุที่ได้จากการเรียกใช้เมธอด getAccount() ของคลาส Member โดยมีตัวแปร manee เป็น this นั่นคือค่าใน maneeAccount จะเป็นวัตถุ new Account(“A111”, 0)
เราก็จะพอมองเห็นว่า เราจะหาค่าของฟิลด์ total ได้ต่อจากจุดนี้
เมื่อตามไปดูนิยามของคอนสตรักเตอร์ของคลาส Account ที่มี 2 พารามิเตอร์ ซึ่งจะต้องเป็น

1. คอนสตรักเตอร์ของคลาส Account เท่านั้น
2. พารามิเตอร์ตัวที่ 1 ต้องเป็นชนิด String เท่านั้น - ดูได้จากค่า “A111” ซึ่งเป็น String
3. พารามิเตอร์ตัวที่ 1 ต้องเป็นชนิด int เท่านั้น - ดูได้จากค่า 0 ซึ่งเป็น int

เมื่อเราไปดูที่คลาส Account ก็จะพบว่ามีคอนสตรักเตอร์ในบรรทัดที่ 5 เป็นไปตามเงื่อนไขของการวิเคราะห์ทั้ง 3 ข้อ
จากนั้นให้ดูในโปรแกรมของคอนสตรักเตอร์ ก็จะพบว่ามีการตั้งค่า this.total = total
เมื่อเราเทียบพารามิเตอร์ ก็จะพบว่า ค่า total เป็นพารามิเตอร์ตัวที่ 2 ซึ่งมีค่าเป็น 0
และมีการตั้งค่านั้นให้กับฟิลด์ this.total

ทำให้สรุปได้ว่าฟิลด์ total ของวัตถุนี้มีค่าเป็น 0

กลับมาที่เมธอด buy(int)
เมื่อรู้แล้วว่าฟิลด์ total (เขียน this.total) มีค่าเป็น 0 จากการไล่โปรแกรม
เราบวกค่า amount เข้าไป และเรารู้ว่าค่า amount เป็น 5 เพราะเป็นค่าของพารามิเตอร์ตัวที่ 1 ของเมธอด buy(int)
ทำให้หลังจากการเรียกใช้โปรแกรมในบรรทัดที่ 8 นี้ ค่าฟิลด์ total ของวัตถุ maneeAccount ก็จะกลายเป็น 0 + 5 = 5 นั่นเอง

บรรทัดที่ 9 สามารถอธิบายในทำนองเดียวกันกับบรรทัดที 8
มองย้อนกลับไปเราจะพบว่าตัวแปรวัตถุ manaAccount เป็น new Account(“A123”, 0) และนั่นหมายถึงฟิลด์ total (เขียนด้วย this.total) ของวัตถุนี้เป็น 0
การเรียกใช้ manaAccount.buy(10) ก็จะทำให้ฟิลด์ total ของวัตถุ manaAccount มีค่าเป็น 0 + 10 = 10

บรรทัดที่ 10 เป็นการเรียกเมธอด getter จากตัวแปรวัตถุ ทั้ง manee และ maneeAccount

```
System.out.println(manee.getName() + " has id: " + maneeAccount.getId() + " (Status:" + manee.getStatus() + ")");
```

โดย manee.getName() คือการเรียกใช้เมธอด getName() ของคลาส Member โดยใช้ตัวแปรวัตถุ manee เป็นตัวแปร this
จากหลักการเดิม เรารู้ว่าเมธอด getName() นี้เป็นเมธอดของคลาส Member เนื่องจากตัวแปร manee มีชนิดเป็น Member
ในทำนองเดียวกัน
maneeAccount.getId() คือการเรียกใช้เมธอด getId() ของคลาส Account โดยใช้ตัวแปรวัตถุ maneeAccount เป็นตัวแปร this
เพราะ maneeAccount มีชนิดเป็นคลาส Account
และ
manee.getStatus() คือการเรียกใช้เมธอด getStatus() ของคลาส Member โดยใช้ manee เป็นตัวแปร this ด้วยเหตุผลเดียวกัน

การเรียกใช้เมธอดทั้ง 3 ถูกกระทำแล้วได้ผลลัพท์กลับมาเป็น String จากนั้นก็ถูกบวกรวมเข้าเป็นประโยค และแสดงผลออกมาด้วยเมธอด println() ของ System.out

บรรทัดที่ 11-13 สามารถอธิบายได้ด้วยใช้แนวทางเดียวกับบรรทัดที่ 10

คลาส Member

```
1 package oot.lab5;

2 public class Member {
3    private String name;
4    private Account account;
5    private String status;
6    public Member(String name, Account account) {
7        super();
8        this.name = name;
9        this.account = account;
10       this.status = "No Status";
11   }
12   public Member(String name, Account account, String status) {
13       super();
14       this.name = name;
15       this.account = account;
16       this.status = status;
17   }
18   public void setName(String name) {
19       this.name = name;
20   }
21   public String getName() {
22       return this.name;
23   }
24   public void setStatus(String status) {
25       this.status = status;
26   }
27   public String getStatus() {
28       return this.status;
29   }
30   public void setAccount(Account account) {
31       this.account = account;
32   }
33   public Account getAccount() {
34       return this.account;
35   }
36}
```

คลาส Account

```
1 package oot.lab5;

2 public class Account {
3    private String id;
4    private int total;

5    public Account(String id, int total) {
6        super();
7        this.id = id;
8        this.total = total;
9    }

10    public void buy(int amount) {
11       this.total = this.total + amount;
12   }

13   public void setId(String id) {
14       this.id = id;
15   }

16   public String getId() {
17       return this.id;
18   }

19   public void setTotal(int total) {
20       this.total = total;
21   }

22   public int getTotal() {
23       return this.total;
24   }
25}
```

ข้อ 2 คลาส Main

```
1 package oot.lab5;

2 public class Main {

3    public static void main(String[] args) {
4        new Pen().sell();
5    }
6 }
```

ข้อนี้เริ่มต้นที่เมธอด main ของคลาส oot.lab5.Main โดยมีโปรแกรมอยู่บรรทัดเดียวคือบรรทัดที่ 4

บรรทัดที่ 4 โปรแกรมทำงานโดยการสร้างวัตถุของคลาส Pen แล้วเรียกใช้เมธอด sell() เลย

บรรทัดนี้อ่านว่า “เรียกใช้เมธอด sell() ของคลาส Pen โดยให้วัตถุ new Pen() เป็นตัวแปร this ตลอดการเรียกใช้เมธอด sell() นี้”

เมื่อดูไปที่การประกาศเมธอด sell() ของคลาส Pen ในบรรทัดที่ 6 พบว่ามีการอ้างถึงฟิลด์ this.type แล้วนำค่าของฟิลด์ดังกล่าวพิมพ์ออกมาด้วย System.out.println()

เราจึงต้องมาดูว่าค่าใน this.type คือค่าอะไร

ปรากฏว่า ไม่มีการประกาศฟิลด์ type ไว้ในคลาส Pen

เทคนิค - การวิเคราะห์

ถ้าหาฟิลด์หรือเมธอดในคลาส ๆ หนึ่งไม่เจอ ให้หาต่อในคลาสแม่ - นี่คือคุณสมบัติการสืบทอดของคลาส (Inheritance)

จากบรรทัดการประกาศคลาส Pen เราพบว่าคลาสนี้ extends มาจากคลาส Product เราก็ไปดูต่อที่คลาส Product

จะพบว่าคลาส Product มีฟิลด์ชื่อ type ประกาศอยู่ เป็นชนิด String

ก็สามารถทำการวิเคราะห์ต่อว่า this.type มีค่าอะไรบรรจุอยู่

จากความหมายของ new Pen().sell() เรารู้ว่า จะมีการใช้ new Pen() เป็นวัตถุ this และในเมธอด sell() มีการอ้างถึง this.type

นั่นคือตัวแปร this ในขณะที่เมธอด sell() ทำงานก็คือจะมีค่าเป็นวัตถุ new Pen() นั่นเอง

ดังนั้นจึงสามารถวิเคราะห์ต่อมายังคอนสตรักเตอร์ของคลาส Pen ได้ (ดูบรรทัดที่ 3 ของคลาส Pen)
ที่บรรทัดที่ 3 ของคลาส Pen เรามีการประกาศคอนสตรักเตอร์ที่ไม่รับพารามิเตอร์ไว้ โดยมีโปรแกรมอยู่ 1 บรรทัดคือ

```
super("Pen");
```

ความหมายของ super(…) คือการเรียกคอนสตรักเตอร์ของคลาสแม่ ในที่นี้คือคลาส Product
และจะมีการส่งค่า “Pen” - เรารู้ว่าเป็น String ไปให้คอนสตรักเตอร์ดังกล่าว

เทคนิค - การใช้ super
ใช้ได้ใน 2 ลักษณะคือ super(…); และ super.<ชื่อเมธอด>(…);
super(…); เอาไว้เรียกใช้คอนสตรักเตอร์ของคลาสแม่
super.<ชื่อเมธอด>(…); เอาไว้เรียกใช้เมธอดของคลาสแม่

เรารู้ว่าต้องไปดูที่คอนสตรักเตอร์ของคลาส Product ที่รับพารามิเตอร์ตัวที่ 1 เป็นชนิด String
ซึ่งอยู่ที่บรรทัดที่ 4 ของคลาส Product
คำว่า “Pen” ที่ถูกส่งเข้ามาก็จะอยู่ในตัวแปรชื่อ type ซึ่งเป็นชื่อของพารามิเตอร์ตัวที่ 1
จากนั้นจะพบว่ามีการตั้งค่า this.type = type โดยการนำค่าในตัวแปร type ไปให้กับฟิลด์ชื่อ type ของวัตถุ this
(อย่าลืมว่า! ตอนนี้ this คือวัตถุ new Pen() จากโปรแกรม main บรรทัดที่ 4)
ทำให้ค่าฟิลด์ this.type มีค่าเป็น “Pen”
รวมถึงคำสั่ง System.out.println() ในคอนสตรักเตอร์ทำงาน

จากนั้นเรากลับมาดูที่เมธอด sell() ของคลาส Pen เราก็จะพบว่าคำสั่ง System.out.println ในเมธอดนี้ก็จะทำงานโดยการพิมพ์ค่า “sell a Pen” ออกมา

คลาส Product

```
1 package oot.lab5;

2 public abstract class Product {
3    protected String type;

4    public Product(String type) {
5        this.type = type;
6        System.out.println("Constructor of " + this.type + " is called.");
7    }

8    public abstract void sell();

9 }
```

คลาส Pen

```
1 package oot.lab5;

2 public class Pen extends Product {

3    public Pen() {
4        super("Pen");
5    }

6    public void sell() {
7        System.out.println("sell a " + this.type);
8    }
9 }
```

ข้อ 3 คลาส Main

```
1 package oot.lab5;

2 public class Main {

3    public static void lead(Team t) {
4        System.out.println(t.getName() + " has captain:" + t.getCaptain());
5    }

6    public static void main(String[] args) {
7        Team team = new Team();
8        England englandTeam = new England();
9        lead(team);
10       lead(englandTeam);
11   }
12}
```

ข้อนี้เริ่มโปรแกรมที่เมธอด main ของคลาส oot.lab5.Main

บรรทัดที่ 7 เป็นการสร้างวัตถุของคลาส Team แล้วเก็บไว้ในตัวแปรชื่อ team - ด้วยหลักการเดิม ให้คิดว่าตัวแปร team เป็นค่า new Team()

บรรทัดที่ 8 เป็นการสร้างวัตถุของคลาส England ไว้ในตัวแปรชื่อ englandTeam - ด้วยหลักการเดิมให้คิดว่าตัวแปร englandTeam เป็นค่า new England()

บรรทัดที่ 9 เป็นการเรียกใช้เมธอด lead(Team) ซึ่งประกาศไว้ที่บรรทัดที่ 3 โดยเมธอด lead(Team) รับพารามิเตอร์ 1 ตัวเป็นชนิดคลาส Team

และจากการประกาศจะเห็นว่าในเมธอดดังกล่าวมีพารามิเตอร์ชื่อ t เป็นพารามิเตอร์ตัวที่ 1 เมื่อเรียกใช้ lead(team); ตามบรรทัดนี้ ค่าในตัวแปร team จึงไปอยู่ที่ตัวแปร t สำหรับใช้ในเมธอด lead(Team)

จากโปรแกรมในเมธอด lead(Team) จะเห็นได้ว่าในบรรทัดที่ 4 มี

- การเรียกใช้เมธอด getName() ของคลาส Team โดยใช้ t เป็นตัวแปร this, และ
- การเรียกใช้เมธอด getCaptain() ของคลาส Team โดยใช้ t เป็นตัวแปร this เช่นกัน

เมื่อดูในการประกาศเมธอด getName() (คลาส Team บรรทัดที่ 13) และ getCaptain (คลาส Team บรรทัดที่ 19) จะเห็นว่าเป็นการคืนค่าของฟิลด์ this.name และ this.captain ตามลำดับ

จากการเรียกใช้ในบรรทัดที่ 9 เรารู้ว่า this จะมีค่าตามวัตถุ t ซึ่งมีค่าวัตถุตามตัวแปร team (บรรทัดที่ 7) และค่าดังกล่าวคือ new Team()

เราจึงสามารถตามวิเคราะห์ค่าต่อไป ด้วยการไปดูคอนสตรักเตอร์ของ คลาส Team
จึงเป็นกรณีง่าย เนื่องจากในคอนสตรักเตอร์มีการตั้งค่าให้ this.name = “Anonymous team”
และ this.caption = “No captain”
ดังนั้นการแสดงผลของบรรทัดที่ 9 นี้จึงเป็นการแสดงผลด้วยค่าดังกล่าวทั้งสองค่า

บรรทัดที่ 10 ในบรรทัดนี้เป็นการเรียกใช้เมธอด lead(Team) คล้ายกับในบรรทัดที่ 9 แต่สิ่งที่ต่างกันออกไปก็คือ เป็นการเรียกใช้ lead(englandTeam); โดยตัวแปร englandTeam ซึ่งมีชนิดเป็น England จะถูกส่งค่าให้พารามิเตอร์ t ซึ่งมีชนิดเป็น Team
แม้ว่าตัวแปรจะเป็นคนละชนิดกัน แต่คลาส England เป็นคลาสลูกของคลาส Team (ดูการประกาศคลาสจะเห็นว่า class England extends Team) จึงทำให้ค่าตัวแปร t เป็นค่าของวัตถุซึ่งอยู่ในตัวแปร englandTeam ได้ ซึ่งก็คือค่า new England() - ด้วยหลักการเดิมให้พิจารณาค่าวัตถุอยู่ในรูปประโยค new แล้วจะทำให้วิเคราะห์ต่อได้

เมื่อมีการเรียกใช้เมธอด getName() ของคลาส Team โดยให้วัตถุของตัวแปร t เป็นตัวแปร this ในโปรแกรมบรรทัดที่ 4
จะเกิดกลไกพิเศษขึ้นเพราะค่าในตัวแปร t ไม่ได้เป็นชนิดคลาส Team
โดยจะมีการแปรสภาพ (morph) ให้ t เป็นคลาสที่ตรงกับค่าวัตถุจริงของมัน - ซึ่งตอนนี้วัตถุมีค่าเป็น new England()
ทำให้เรียกใช้เมธอด getName() ของคลาส England แทนที่จะเป็นการเรียกใช้ปกติ
และแน่นอนว่า ตัวแปร this ที่จะใช้ในการเรียกใช้เมธอด getName() ก็จะเป็นค่า new England()

เทคนิค - การวิเคราะห์คุณสมบัติ Polymorphism

1. วัตถุที่จะใช้เป็นคลาสลูก
2. แต่วัตถุนั้นถูกอ้างอิงอยู่ในรูปของคลาสแม่
3. การเรียกใช้เมธอดหรือฟิลด์บนวัตถุนั้น จะใช้ค่าวัตถุจริง ซึ่งเป็นคลาสลูกเสมอ

เมื่อวิเคราะห์ต่อด้วย this เป็นค่า new England() เราก็ต้องพิจารณาคอนสตรักเตอร์ของคลาส England

ซึ่งจะเห็นว่า this.name ถูกตั้งค่าเป็น “England” ทำให้เมธอด getName() ในบรรทัดที่ 4 คืนค่าเป็น “England” ออกมา

ในทำนองเดียวกัน การเรียกใช้เมธอด getCaptain() ของคลาส Team ก็ต้องมีการปรับตามกลไก Polymorphism

ทำให้เป็นการเรียกใช้เมธอด getCaption() ของคลาส England และให้ t เป็นตัวแปร this แทน

ผลลัพธ์ของบรรทัดที่ 4 จึงเป็นการแสดงผล

England has caption: Rio Ferdinand

คลาส Team

```
1 package oot.lab5;

2 public class Team {
3    protected String name;
4    protected String captain;

5    public Team() {
6        this.name = "Anonymous team";
7        this.captain = "No captain";
8    }
9
10   public void setName(String name) {
11       this.name = name;
12   }

13   public String getName() {
14       return this.name;
15   }

16   public void setCaptain(String captain) {
17       this.captain = captain;
18   }

19   public String getCaptain() {
20       return this.captain;
21   }
22}
```

คลาส England

```
1 package oot.lab5;

2 public class England extends Team {
3    public England() {
4        super();
5        this.name = "England";
6        this.captain = "Rio Ferdinand";
7    }
8}
```