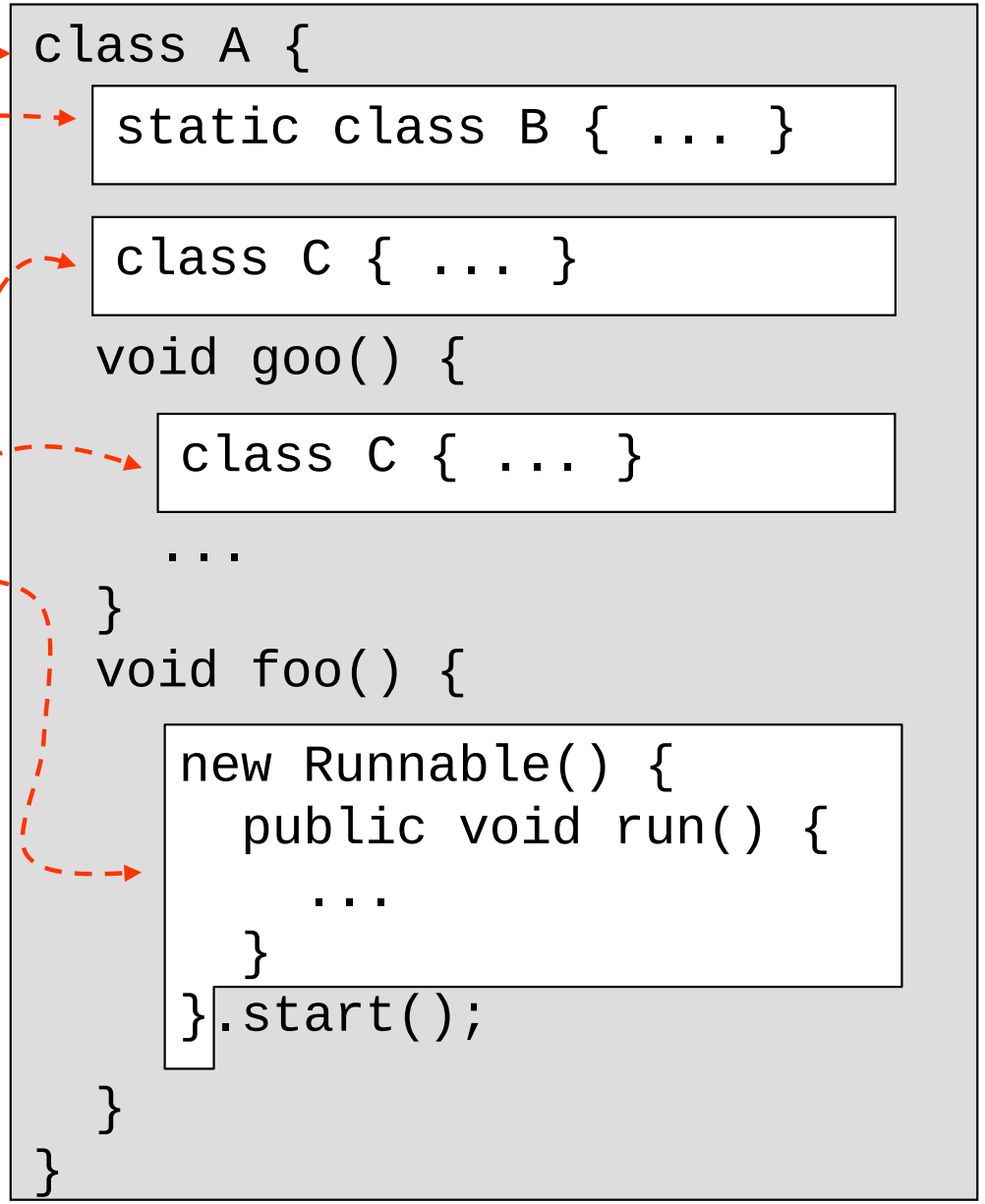
ภาษาจาวา – Nested Classes

สมชาย ประสิทธิ์จูตระกูล

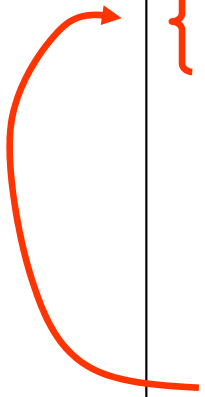
Classes

- Top level classes
- Nested classes
  - Nested top-level
  - Member inner
  - Local inner
  - Anonymous inner



Nested top-level classes

- อยู่ใน top-level หรือ nested top-level
- มี static กำกับ class
- อ้างอิง static members ของ outer class ได้เท่านั้น
- มีสมาชิกได้ทั้งแบบ static และ non-static

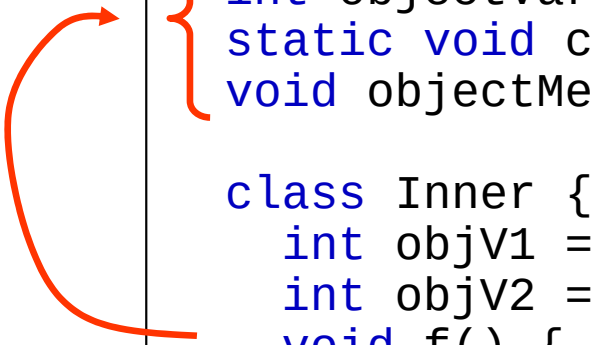


```
class Outer {
  static int classVar;
  static void classMethod() {}
  int objectVar;
  void objectMethod() {}

  static class NTL {
    static int cV = classVar;
    int objV = classVar;
    static void f() { classMethod(); }
    void g() { classMethod(); }

    static class NTL2 { ... }
  }
}
```

```
Outer a = new Outer();
Outer.NTL b = new Outer.NTL();
```

Member inner classes

- อยู่ใน top-level, nested top-level, member inner
- ไม่มี static กำกับ class
- อ้างอิงทุก ๆ members ของ outer class ได้
- มีสมาชิกได้เฉพาะแบบ non-static



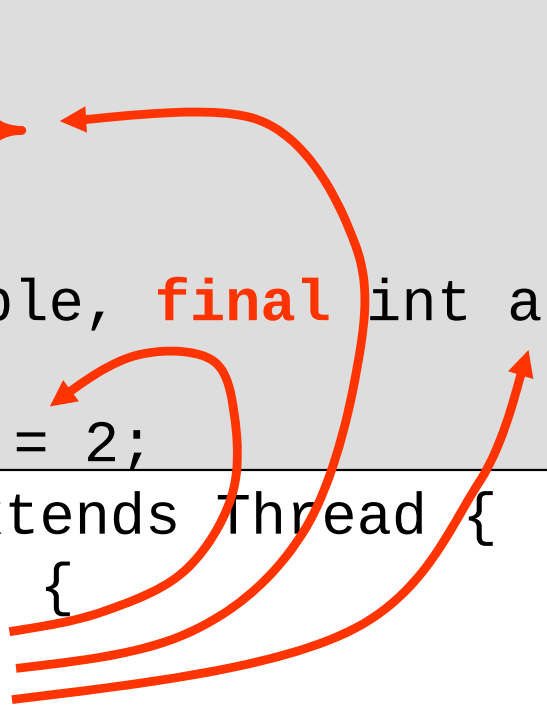
```
class Outer {
  static int classVar;
  int objectVar;
  static void classMethod() {}
  void objectMethod() {}

  class Inner {
    int objV1 = classVar;
    int objV2 = objectVar;
    void f() { classMethod(); }
    void g() { objectMethod(); }
  }
  Inner f() { return new Inner(); }
}
```

```
Outer a = new Outer();
Outer.Inner b;
b = a.new Inner();
b = new Outer().new Inner();
```

ต้องมี outer object ถึงจะมี member inner object ได้

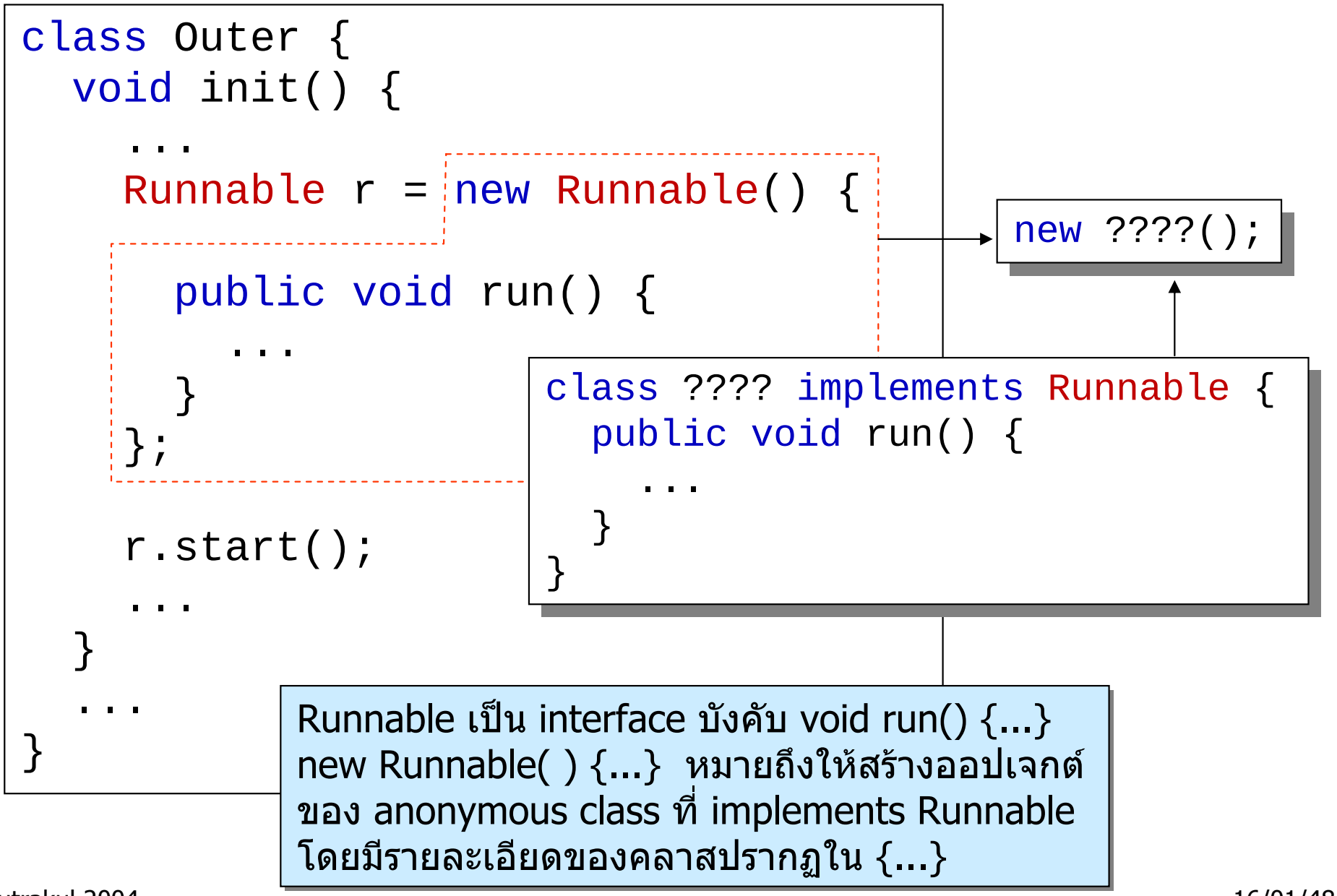
Local Inner

```
class Outer {
  static int c;
  static void f(){}
  int v;
  void g(){}
  void init(boolean enable, final int a) {
    String s = “abc”;
    final int aLocalVar = 2;
    class LocalInner extends Thread {
      public void run() {
        ...
      }
    }
    new LocalInner().start();
  }
}
```

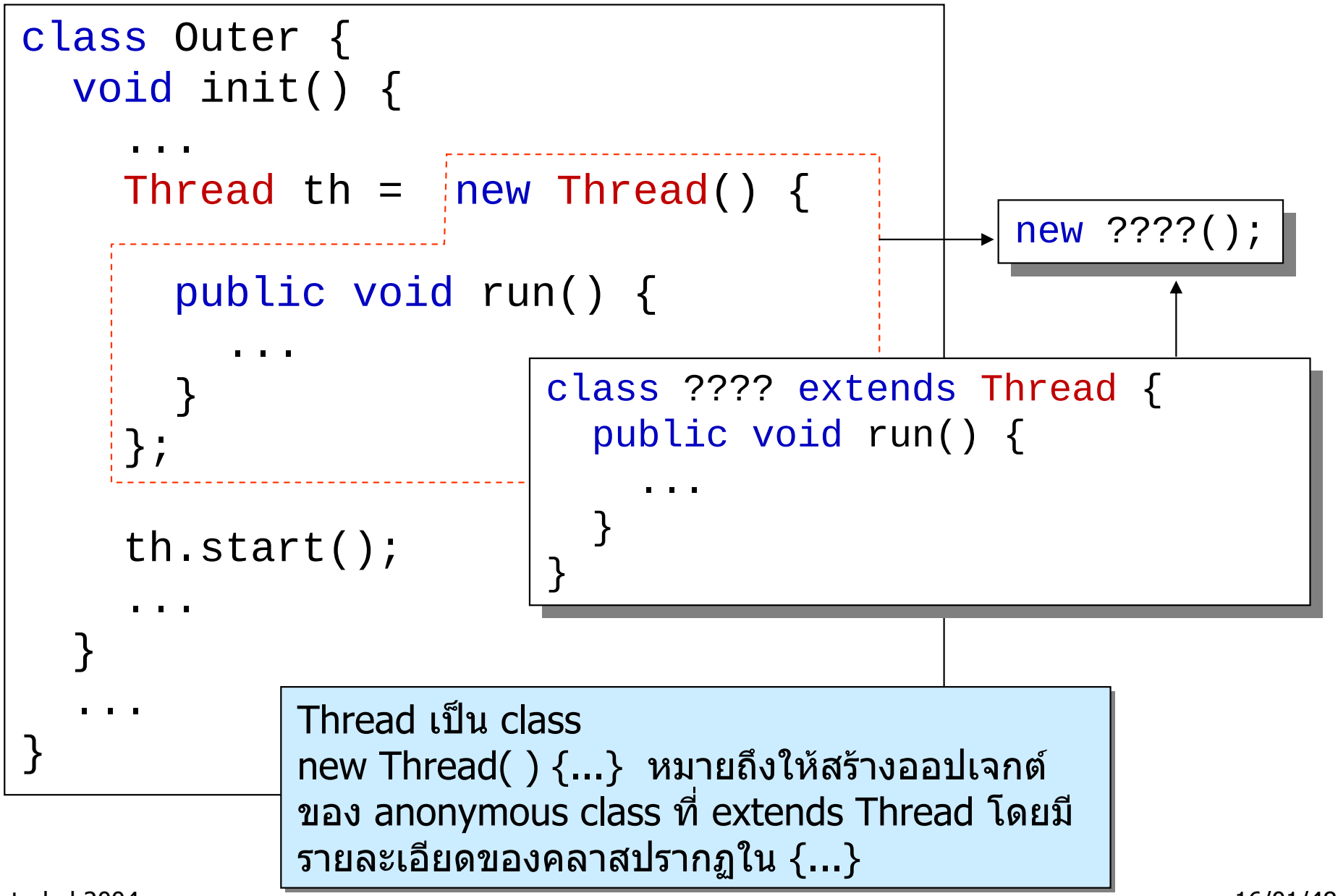


Local inner class คือ member inner class ที่เขียนนิยามใน method (อ้างอิง local var. ที่เป็น final ได้)

Anonymous Inner : interface



Anonymous Inner : class or abstract class

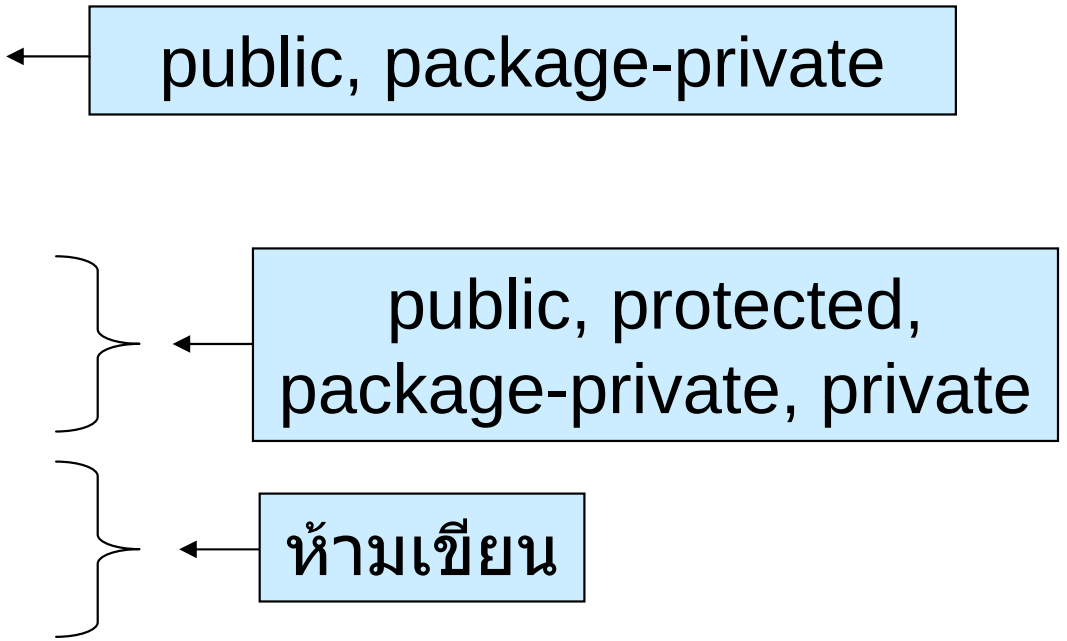


Anonymous local inner classes

- คือ local inner classess อย่างหนึ่ง
- เขียน บรรยายคลาส + สร้างออปเจกต์ พร้อมกัน
- เขียนบรรยาย
  - คลาสใหม่ที่ implements interface ที่รู้จัก
    พร้อมเขียนทุก ๆ เมท็อดที่ interface นั้นบังคับ
  - คลาสใหม่ที่ extends ของคลาสอื่น
    เพื่อ override บางเมท็อดของคลาสพ่อ
- ใช้ออปเจกต์ของ anonymous คลาส
  - ในรูปของ interface ที่ implements หรือ
  - ในรูปของคลาสพ่อที่ extends
- anonymous = ไม่มีชื่อ เขียน constructor ไม่ได้
- anonymous class ควรมีขนาดเล็ก ๆ

Access Controls

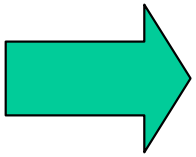
- Top level class
- Nested class
  - Nested top-level
  - Member inner
  - Local inner
  - Anonymous inner



ใช้ nested top-level class เมื่อไร ?

```
class A {
   ...
   B b = new B();
   ...
}
```

```
class B {
   ...
}
```



```
class A {
   ...
   B b = new B();
   ...

   private static
   class B {
      ...
   }

}
```

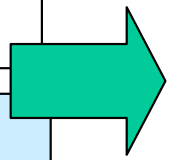
คลาส B ถูกใช้เฉพาะในคลาส A เท่านั้น ก็ให้ B เป็น nested top-level ซึ่ง private เฉพาะในคลาส A

คลาสที่มีความสัมพันธ์กันควรเขียนอยู่ใกล้กัน

ใช้ member inner class เมื่อไร ?

```
class Bag {
 Object [] data;
 Itr iterator() {
  return new Itr(this);
 }
 ...
}
```

```
class Itr {
  private Bag bag;
  private int idx;
  Itr(Bag b) {
   this.bag = b;
  }
  Object next() {
    return bag.data[idx++];
  }
  ...
}
```



```
class Bag {
 private Object [] data;
 Itr iterator() {
  return new Itr();
 }
  class Itr {
    private int idx;
    Object next() {
      return bag.data[idx++];
    }
    ...
  }
  ...
}
```

ลดความซับซ้อน : member inner ใช้ members ของ outer ได้เลย

ใช้ local inner class เมื่อไร ?

- เขียนเอง ใช้เอง ภายในเมท็อด ข้างนอกไม่รู้จัก
- มักใช้ออปเจกต์ของ local คลาสในรูปของคลาสอื่น

```
static Point getRectilinearPoint(Point p) {
  class RPoint extends Point implements Comparable {
    RPoint(int x, int y) {
      super(x, y);
    }
    public int compareTo(Object obj) {
      Point that = (Point) obj;
      int dThis = Math.abs(this.x) + Math.abs(this.y);
      int dThat = Math.abs(that.x) + Math.abs(that.x);
      return dThis - dThat;
    }
  }
  return new RPoint(p.x, p.y);
}
```

ใช้ anonymous class เมื่อไร ?

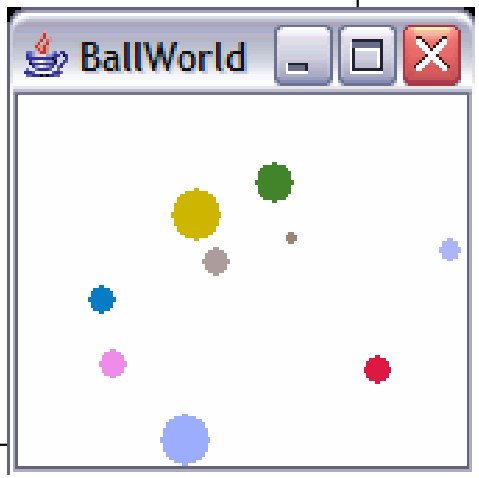
```
Point getRectilinearPoint(Point p) {
  class RPoint extends Point implements Comparable {
    RPoint(int x, int y) { super(x, y); }
    public int compareTo(Object obj) {
      Point that = (Point) obj;
      int dThis = Math.abs(this.x) + Math.abs(this.y);
      int dThat = Math.abs(that.x) + Math.abs(that.x);
      return dThis - dThat;
    }
  }
  return new RPoint(p.x, p.y);
}
```

```
Point getRectilinearPoint(final Point p) {
  return new Point() // class ??? extends Point
  {
    { x = p.x; y = p.y; } // initializer block
    public int compareTo(Object obj) {
      Point that = (Point) obj;
      int dThis = Math.abs(this.x) + Math.abs(this.y);
      int dThat = Math.abs(that.x) + Math.abs(that.x);
      return dThis - dThat;
    }
  };
}
```

local inner ที่ใช้เพียงตำแหน่งเดียวเท่านั้นในเมท็อด → anonymous

Lab 6 : BallWorld

```
public class BallWorld extends Canvas {
  Ball[] balls;
  public void begin() {
    for (int i = 0; i < 10; i++) {
      balls[i] = new Ball(this);
    }
    Graphics g = getGraphics();
    while (true) {
      for (int i = 0; i < balls.length; i++) {
        balls[i].move(); balls[i].draw(g);
      }
      ...
      repaint();
    }
  }
  ...
```



Lab 6 : Balls

```
class Ball {
  int x, y, r, dx, dy;
  BallWorld world;
  public Ball(BallWorld ballWorld) {
    world = ballWorld;
    ...
  }
  public void move() {
    int ww = world.getWidth();
    x = x + dx
    int d = 2 * r;
    if (x + d >= ww) { x = ww - d; dx = -dx; }
    if (x <= 0) { x = 0; dx = -dx; }
    ...
  }
  ...
}
```

Lab 6.1 : Nested Top-Level Class

- ภายใต้ package com.somchai.lab6.nested มีคลาส Ball และ BallWorld ที่เป็น top-level ทั้งคู่
- จงทำคลาส Ball ให้เป็น private nested top-level เพื่อใส่ไว้ภายใน BallWorld

Lab 6.2 : Member Inner Class

- ภายใต้ package com.somchai.lab6.member มีคลาส Ball และ BallWorld ที่เป็น top-level ทั้งคู่
- จงทำคลาส Ball ให้เป็น private member inner เพื่อใส่ไว้ภายใน BallWorld

Lab 6.3 : Local Inner Class

- ภายใต้ package com.somchai.lab6.local มีคลาส Ball และ BallWorld ที่เป็น top-level ทั้งคู่
- จงทำคลาส Ball ให้เป็น private local inner class เพื่อใส่ไว้ภายในเมท็อด begin

Lab 6.4 : Anonymous Local Inner Class

- ภายใต้ package com.somchai.lab6.anonymous มีคลาส Ball และ BallWorld ที่เป็น top-level ทั้งคู่
- ลองพยายามทำคลาส Ball ให้เป็น anonymous inner เพื่อใส่ไว้ภายในเมท็อด begin

Lab 6.5 : Anonymous Local Inner Class

- ภายใต้ package com.somchai.lab6.comparator มีคลาส Rational ที่ implements Comparable และเขียน compareTo ไว้เรียบร้อยแล้ว
- จงเขียน static method ชื่อ reverseOrder() ในคลาส Rational ที่คืน Comparator object ซึ่งเมื่อใช้กับ Arrays.sort(...) จะเรียงลำดับจากมากไปน้อย

```
public class Rational {
   ...
   public static Comparator reverseOrder() {
      ...
   }
}
```

Lab 6.5 : Anonymous Local Inner Class

- Comparator เป็น interface ที่บังคับเมท็อด
  - public int compare(Object obj1, Object obj2)
  - compare ใช้ในการจัดอันดับของออปเจกต์จากซ้ายไปขวา
  - compare คืนค่าในทำนองเดียวกับ compareTo
    - คืนค่า 0 แสดงว่า obj1 มีค่าเท่ากับ obj2
    - คืนค่า < 0 แสดงว่า obj1 ต้องอยู่ทางซ้ายของ obj2
    - คืนค่า > 0 แสดงว่า obj1 ต้องอยู่ทางขวาของ obj2
- นำ comparator ไปใช้กับ Arrays.sort ได้
  - Arrays.sort(Object[] d) แบบนี้ไม่ระบุ comparator จะใช้เมท็อด compareTo ของตัวออปเจกต์เอง
  - Arrays.sort(Object[] d, Comparator c) แบบนี้ระบุ comparator จะใช้ compare ของ comparator และไม่สนใจ compareTo ของออปเจกต์

Lab 6.5 : Anonymous Local Inner Class

```
class ReverseComparator implements Comparator {
  public int compare(Object obj1, Object obj2) {
    Comparable c = (Comparable) obj1;
    return -c.compareTo(obj2); // reverse order
  }
}
```

ตัวอย่าง reverse comparator แบบเขียนเป็น top-level คลาส
พร้อมตัวอย่างการทดสอบ

```
class SortRationalTest {
  public static void main(String[] a) {
    Rational[] r = {
      new Rational(2, 1), new Rational(2, 2),
      new Rational(2, 3), new Rational(30, 4),
      new Rational(30, 3), new Rational(30, 2)
    };
    Arrays.sort(r, new ReverseComparator());
    System.out.println(Arrays.asList(r));
  }
}
```