จาวา – Access Control

สมชาย ประสิทธิ์จูตระกูล

Package

- ระบบจัดเก็บคลาสต่างๆ ในรูปของ package
- ชื่อ package : คำ ๆ คั่นด้วยจุด
- ชื่อเต็มของคลาสคือ *ชื่อ package* . *ชื่อคลาส*
- มักเริ่มชื่อ package ด้วย domain name ขององค์กร แต่เขียนแบบกลับด้าน เพื่อป้องกันไม่ให้ซ้ำ

```
package com.somchai;
class Demo1 {
   ...
}
```

```
package com.somchai;
class Demo2 {
   ...
}
```

```
package com.somchai.games;
class DiffSpot {
   ...
}
```

```
package com.somchai.games;
class MemoryGame {
   ...
}
```

การอ้างอิงคลาสนอก package

- เมื่อต้องการใช้คลาสที่อยู่นอก package ของตัวเอง
  - เขียนชื่อ class รวมทั้ง package ให้ครบถ้วน
  - ใช้ import
- ระบบ `import java.lang.*;` ให้อัตโนมัติ

```
package com.somboon;
public class Main {
  public static void main(String[] a) {
    new com.somchai.games.MemoryGame().begin();
  }
}
```

```
package com.somboon;
import com.somchai.games.MemoryGame;
public class Main {
  public static void main(String[] a) {
    new MemoryGame().begin();
  }
}
```

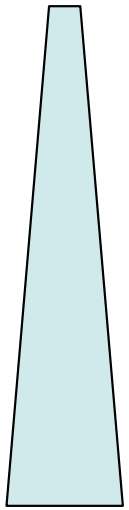
การ import ทุกคลาสของ package

```
package com.somboon;
import com.somchai.games.MemoryGame;
import com.somchai.games.DiffSpot;

public class Main {
  public static void main(String[] a) {
    new MemoryGame().begin();
    new DiffSpot().begin();
  }
}
```

```
package com.somboon;
import com.somchai.games.*;

public class Main {
  public static void main(String[] a) {
    new MemoryGame().begin();
    new DiffSpot().begin();
  }
}
```

Access Controls

- เราสามารถจำกัดขอบเขตว่าคลาสใดมาใช้ class, methods, และ fields ต่าง ๆ ของคลาสเรา ได้สี่ประเภท
  - private : อนุญาตให้ใช้เฉพาะภายในคลาสตัวเอง
  - *package-private* : เฉพาะคลาสใน package เดียวกัน
  - protected : อนุญาตเฉพาะคลาสลูกหลาน หรือคลาสใน package เดียวกัน
  - public : อนุญาตทุก ๆ คลาส



ใส่ public, protected, private นำหน้าชื่อ class, method, และ field ในกรณีไม่ใส่ จะเป็น access control แบบ *package-private*

Access Control

```
package a.a.b;
import a.a.a.A;
public class B {
  void m() {
    A a = new A();
    a.f1 = 3; // x
    a.f2 = 3; // x
    a.f3 = 4; // x
    a.f4 = 1;
    a.m1();   // x
    a.m2();   // x
    a.m3();   // x
    a.m4();
  }
}
```

```
package a.a.a;
public class A {
  private int f1;
  int f2;
  protected int f3;
  public int f4;
  private void m1() {}
  void m2() {}
  protected void m3() {}
  public void m4() {}
}
```

Access Control

```
package a.a.b;
import a.a.a.A;
public class B extends A {
  void m() {
    A a = new A();
    a.f1 = 3; // x
    a.f2 = 3; // x
    a.f3 = 4;
    a.f4 = 1;
    a.m1();   // x
    a.m2();   // x
    a.m3();
    a.m4();
  }
}
```

```
package a.a.a;
public class A {
  private int f1;
  int f2;
  protected int f3;
  public int f4;
  private void m1() {}
  void m2() {}
  protected void m3() {}
  public void m4() {}
}
```

Access Control

```
package a.a.a;

public class B {
  void m() {
    A a = new A();
    a.f1 = 3; // x
    a.f2 = 3;
    a.f3 = 4;
    a.f4 = 1;
    a.m1();   // x
    a.m2();
    a.m3();
    a.m4();
  }
}
```

```
package a.a.a;
public class A {
  private int f1;
  int f2;
  protected int f3;
  public int f4;
  private void m1() {}
  void m2() {}
  protected void m3() {}
  public void m4() {}
}
```

ไม่ตรวจสอบ access control ตอน run-time

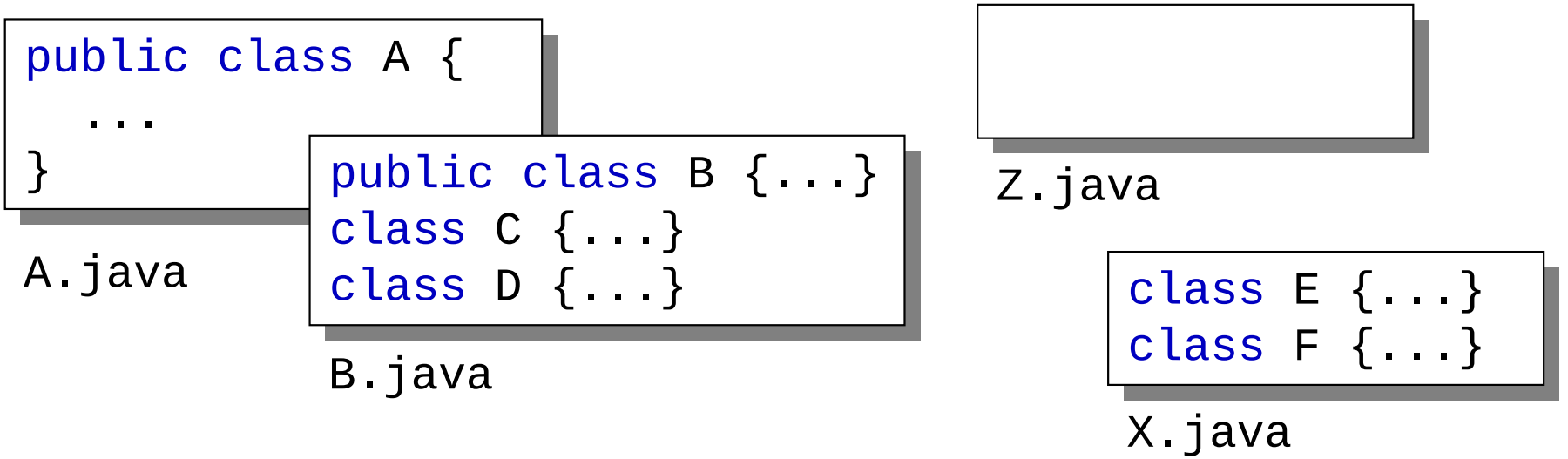
```
package b.b.b;
import a.a.a;
public class Main {
  public static void main(String[] a) {
    Animal.getInstance().say(); // woof woof
  }
}
```

```
package a.a.a;
public class Animal {
 static Animal getInstance() {
   return new Dog();
 }
 void say() { }
}
```

```
package a.a.a;
class Dog extends Animal {
  void say() {
    System.out.println("woof woof");
  }
}
```

Access Control : Class

- กำกับคลาสได้เฉพาะ public หรือ *package-private* (ยกเว้น nested class)
- source code หนึ่งแฟ้ม (.java) เขียนบรรยายได้หลายๆ คลาส แต่มีได้เพียงหนึ่ง public คลาสเท่านั้น และต้องมีชื่อเดียวกับชื่อแฟ้ม



Access Control : Overriding

- private methods ไม่ตกทอดให้ลูกหลาน
- overriding method ของคลาสลูกต้องมี access control ไม่แคบกว่า overridden method ของพ่อ

```
class A {
  public void a() {}
  protected void b() {}
  void c() {}
  private void d() {}
}
```

```
class B extends A {
  protected void a() {}
  void b() {}
  private void c() {}
  void d() {super.d();}
}
```

```
class C extends A {
  public void a() {}
  public void b() {}
  protected void c() {}
  void d() {}
}
```

Constructor

- ป้องกันไม่ให้ผู้อื่นมาสร้าง object ด้วยการ new

คุมกำเนิด

```
package java.awt.color;
public class ColorSpace {
  private static ColorSpace sRGBspace;
  protected ColorSpace(int type, int numcomponents ) { ... }
  public static ColorSpace getInstance(int colorspace) {
    ColorSpace cs;
    switch (colorspace) {
      case CS_sRGB :
       if (sRGBspace == null) {
        sRGBspace = new ICC_ColorSpace(...);
       }
       cs = sRGBspace;
       break;
    }
    ...
    return cs;
  }
  ...
```

เป็นหมัน

```
public final class Math {
  private Math() {}
  public static double random() {...}
  public static double sin(double r) {...}
  public static double cos(double r) {...}
  ...
}
```

ข้อแนะนำ

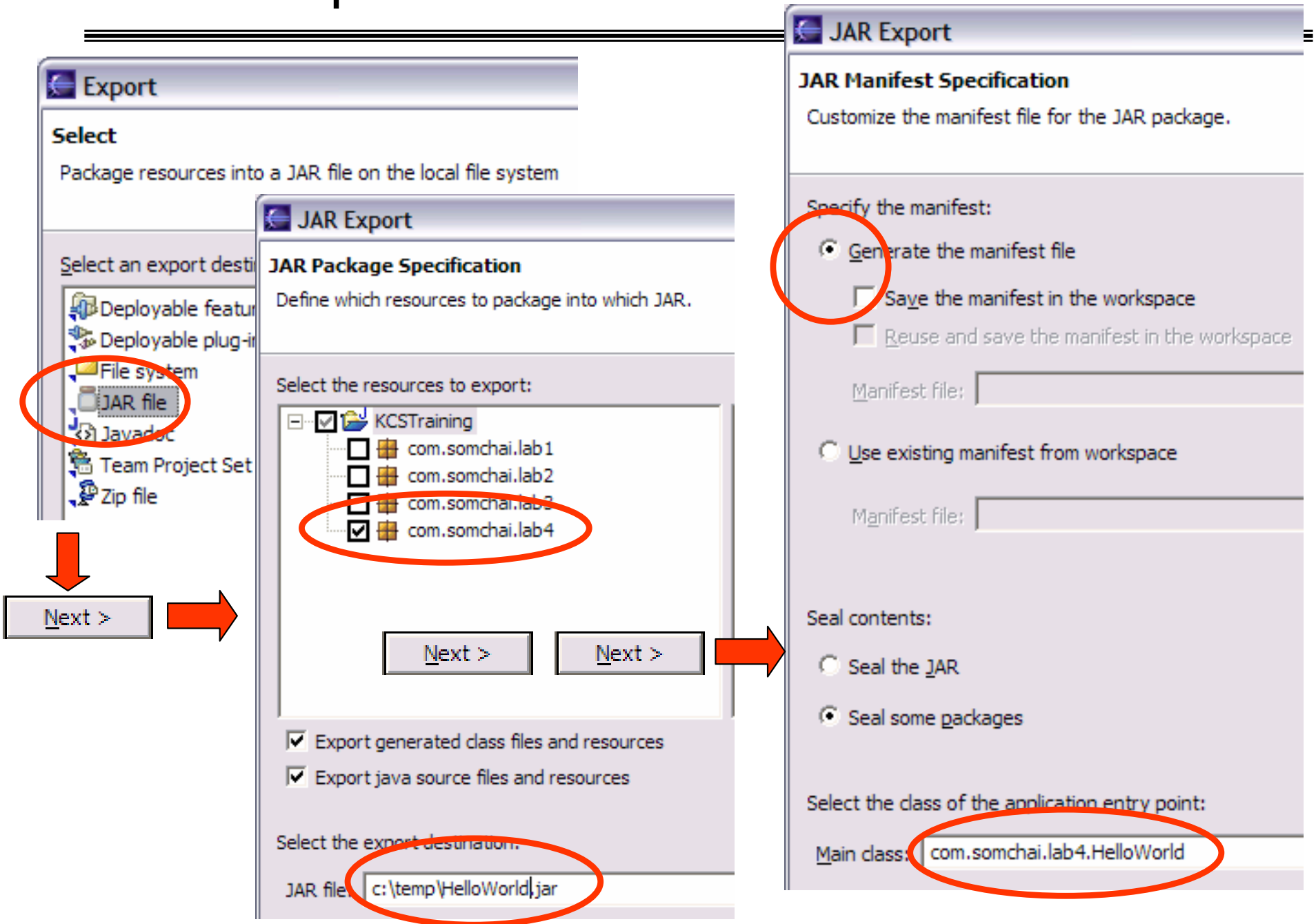
- ปิดปัง (private) ให้มากที่สุด
  - เขียนเอง ควบคุมเอง ใช้เอง เปลี่ยนแปลงได้อย่างอิสระ
- public : มั่นใจว่าต้องการให้บริการทุกคน
- protected : ไม่อยาก public แต่ subclass น่าจะต้องการใช้
- package-private : อยากเป็น private แต่คลาสร่วม package น่าจะต้องการใช้
- ยิ่งเปิดเผยมาก ยิ่งเปลี่ยนแปลงยาก
- fields ทั้งหลายควรเป็น private (ยกเว้น final)

Lab 4 : Jar

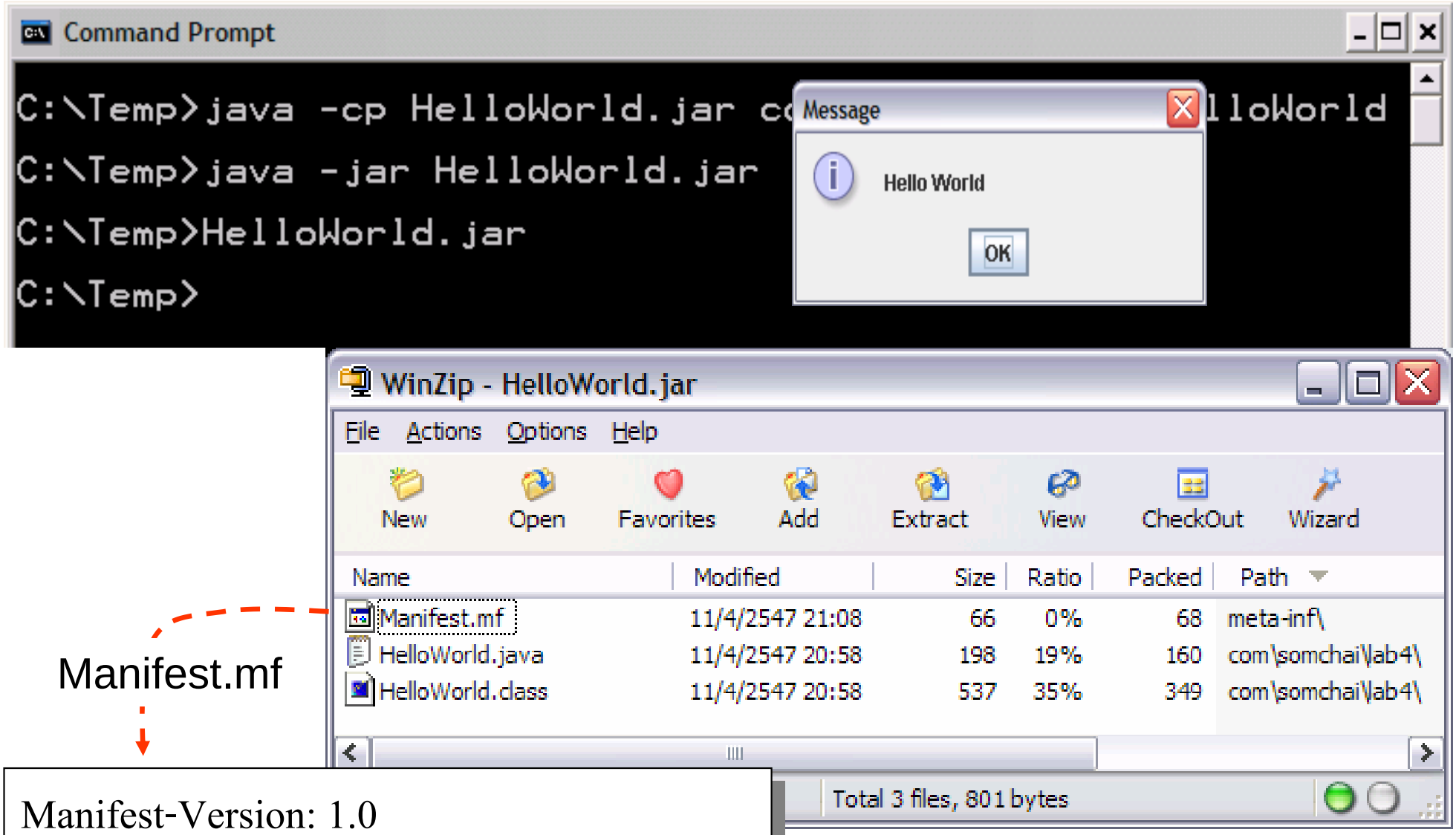
```
package com.somchai.lab4;

import javax.swing.JOptionPane;

public class HelloWorld {
  public static void main(String[] args) {
    JOptionPane.showMessageDialog(null, "Hello World");
  }
}
```

- สร้างคลาส HelloWorld ใน com.somchai.lab4
- สร้าง HelloWorld.jar ตามขั้นตอนในหน้าถัดไป

File -> Export ...


Export
Select
Package resources into a JAR file on the local file system
Select an export desti
Deployable featur
Deployable plug-in
File system
JAR file
Javadoc
Team Project Set
Zip file
Next >
JAR Export
JAR Package Specification
Define which resources to package into which JAR.
Select the resources to export:
KCSTraining
com.somchai.lab1
com.somchai.lab2
com.somchai.lab3
com.somchai.lab4
Next >
Next >
Export generated class files and resources
Export java source files and resources
Select the export destination:
JAR file:
c:\temp\HelloWorld.jar
JAR Export
JAR Manifest Specification
Customize the manifest file for the JAR package.
Specify the manifest:
Generate the manifest file
Save the manifest in the workspace
Reuse and save the manifest in the workspace
Manifest file:
Use existing manifest from workspace
Manifest file:
Seal contents:
Seal the JAR
Seal some packages
Select the class of the application entry point:
Main class:
com.somchai.lab4.HelloWorld

Launching Application from a Jar File



Lab 4.1a : toString

- toString มีไว้ให้เรียกกับออปเจกต์หนึ่ง เมื่อต้องการ string ที่แทนค่าของออปเจกต์นั้น
- ถ้า a เป็นออปเจกต์ แล้วนำ a ไป + กับ string ระบบจะเรียก a.toString( ) ให้อัตโนมัติ
- ถ้า a เป็นออปเจกต์ แล้วเรียก System.out.println(a) ระบบจะเรียก a.toString( )
- ลองเพิ่ม code ข้างล่างนี้ใน comp.somchai.lab4.Rational แล้วสั่งทำงาน

```
public static void main(String[] args) {
  Rational r1 = new Rational(2, 3);
  String s = "r1 = " + r1;
  System.out.println(s);
}
```

Lab 4.1b : toString

- จงเพิ่มเมท็อด public String toString( ) ใน คลาส com.somchai.lab4.Rational เพื่อคืน string ที่อ่านรู้เรื่อง เช่น

```
Rational r = new Rational(2, 3);
System.out.println("r = " + r);
```

จะได้

```
r = 2/3
```

Lab 4.2a : equals

- equals เป็น method เพื่อการเปรียบเทียบว่าออปเจกต์สองตัว "เหมือนกัน" หรือไม่
- ลอง Run com.somchai.lab4.Rational
  - จะได้ false, false, false, false
  - น่าจะได้ false, true, true, true

```
public static void main(String[] args) {
  Rational r1 = new Rational(2, 3);
  Rational r2 = new Rational(2, 3);
  System.out.println(r1 == r2);
  System.out.println(r1.equals(r2));
  System.out.println(r1.equals(new Rational(2,3)));
  System.out.println(r1.equals(new Rational(4,6)));
}
```

Lab 4.2b : equals

- คลาส Object มี

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

- Rational ไม่ได้ override equals
- จงสร้างคลาสใหม่ `com.somchai.lab4.Rational0` ที่ extends จาก `com.somchai.lab4.Rational`
  ซึ่งมี `equals` ที่เปรียบเทียบว่ามี "ค่า" เท่ากันหรือไม่

Lab 4.2b : equals

- ที่คลาส com.somchai.lab4.Rational จงเปลี่ยน
  - ให้ตัวคลาสเป็น public
  - ให้ fields ต่าง ๆ เป็น protected
  - ให้ constructors, reciprocate, add, multiply เป็น public
  - ให้ simplify เป็น private, gcd เป็น protected
- จากนั้นสร้างคลาสใหม่ com.somchai.lab4.Rational0 ในหน้าถัดไป
- Run Rational0 แล้วอธิบายว่าทำไมได้ผลเช่นนั้น

```java
package com.somchai.lab4;

public class Rational0 extends Rational {
  public Rational0() {}
  public Rational0(int n, int d) {
    super(n,d);
  }
  public boolean equals(Object obj) {
    if (! (obj instanceof Rational)) return false;
    Rational r = (Rational) obj;
    return r.numerator == numerator &&
           r.denominator == denominator;
  }
  public static void main(String[] args) {
    Rational r1 = new Rational0(2, 3);
    Rational r2 = new Rational0(4, 6);
    Rational r3 = new Rational(4, 6);
    System.out.println(r1.equals(r2));
    System.out.println(r1.equals(r3));
    System.out.println(r3.equals(r1));
  }
}
```