วาจาจาวา – ตอนที่ ๑
การสร้างออปเจกต์

สมชาย ประสิทธิ์จูตระกูล

การสร้างออปเจกต์

- เขียน constructor ให้ผู้ใช้เรียก new
  - **`BigInteger bi = new BigInteger(128, 30, rand);`**
  - **`Boolean b = new Boolean(t);`**
- เขียน static factory method ให้ผู้ใช้เรียกเมท็อด
  - **`BigInteger bi = BigInteger.probablePrime(128, rand);`**
  - **`Boolean b = Boolean.valueOf(t);`**

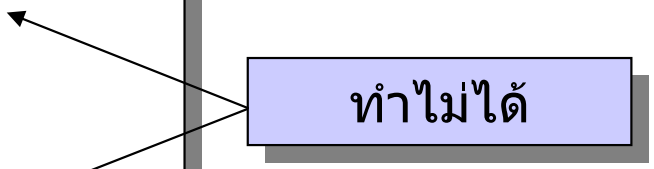
```
public final class Boolean {
  public static final Boolean TRUE = new Boolean(true);
  public static final Boolean FALSE = new Boolean(false);
  private final boolean value;
  public Boolean(boolean value) {
    this.value = value;
  }
  public static Boolean valueOf(boolean b) {
    return (b ? TRUE : FALSE);
  }
  ...
}
```

Constructor

Static factory method

Static factory : constructor ทำไม่ได้

```
public class Complex {
  private final double real, imaginary;
  public Complex(double re, double im) {
    this.real = re; this.imaginary = im;
  }
  public Complex(double r, double theta) {
    this.real = r * Math.cos(theta);
    this.imaginary = r * Math.sin(theta);
  }
}
```




```
public class Complex {
  private final double real, imaginary;
  private Complex(double re, double im) {
    this.real = re; this.imaginary = im;
  }
  public static Complex valueOf(double re, double im) {
    return new Complex(re, im);
  }
  public static Complex valueOfPolar(double r, double theta) {
    return new Complex(r * Math.cos(theta),
                       r * Math.sin(theta));
  }
  ...
```

Static factory : ตั้งชื่อสื่อความหมายได้

```
import java.math.BigInteger;
import java.util.Random;

public class Main {
  public static void main(String[] args) {
    Random rnd = new Random();

    BigInteger bi1 = new BigInteger(128, 30, rnd);
    System.out.println(bi1);

    BigInteger bi2 = BigInteger.probablePrime(128, rnd);
    System.out.println(bi2);
  }
}
```

```
JLab>java Main
325072299559712516298826480793314457969
224772172783110242121431675012695788601
JLab>
```

Static factory : ตั้งชื่อสื่อความหมายได้

```
public class Box {
  public Box(Box b) {...}
  public Box(Box b, double angle) {...}
  public Box(double scale, Box b) {...}
  ...
}
```


```
public class Box {
  public Box(Box b) {...}
  public static Box newInstanceScaled(Box b, double scale) {
    return new Box(b).scale(scale);
  }
  public static Box newInstanceRotated(Box b, double angle) {
    return new Box(b).rotate(angle);
  }
  ...
}
```

Static factory : คุมกำเนิดได้

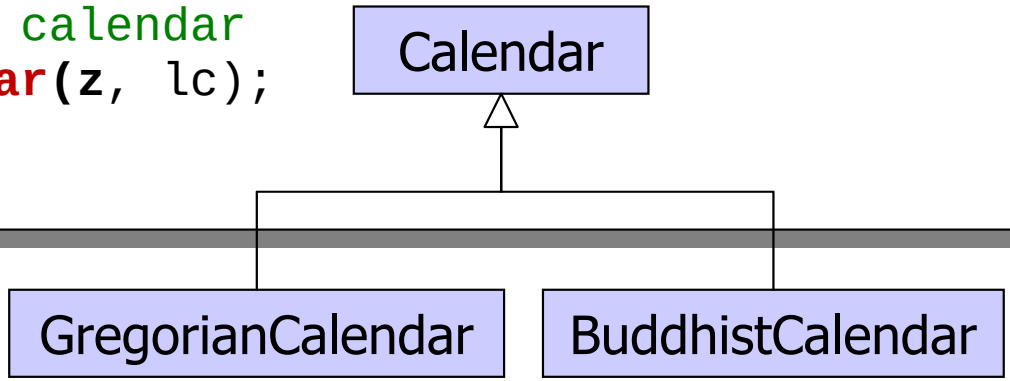
```
public final class Boolean {
  public static final Boolean TRUE = new Boolean(true);
  public static final Boolean FALSE = new Boolean(false);
  private final boolean value;
  public Boolean(boolean value) {
    this.value = value;
  }
  public static Boolean valueOf(boolean b) {
    return (b ? TRUE : FALSE);
  }
  ...
}
```

ต้อง immutable

```
public class Main {
  public static void main(String[] args) {
    boolean b = true;
    Boolean b1 = new Boolean(b);
    Boolean b2 = new Boolean(b);
    Boolean b3 = Boolean.valueOf(b);
    Boolean b4 = Boolean.valueOf(b);
    System.out.println(b1 == b2); // false
    System.out.println(b3 == b4); // true
  }
}
```

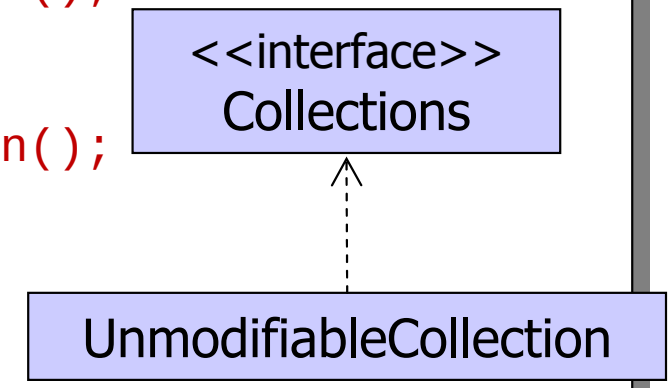
Static factory : สร้างออปเจกต์ของ subtype ได้

```
public abstract class Calendar {
  ...
  protected Calendar() {...}

  public static Calendar getInstance(TimeZone z, Locale lc) {
    return createCalendar(z, lc);
  }
  private static Calendar createCalendar(TimeZone z, Locale lc) {
    // If the specified locale is a Thai locale,
    // returns a BuddhistCalendar instance.
    if ("th".equals(lc.getLanguage())
        && ("TH".equals(lc.getCountry()))) {
      return new sun.util.BuddhistCalendar(z, lc);
    }
    // else create the default calendar
    return new GregorianCalendar(z, lc);
  }
}
```



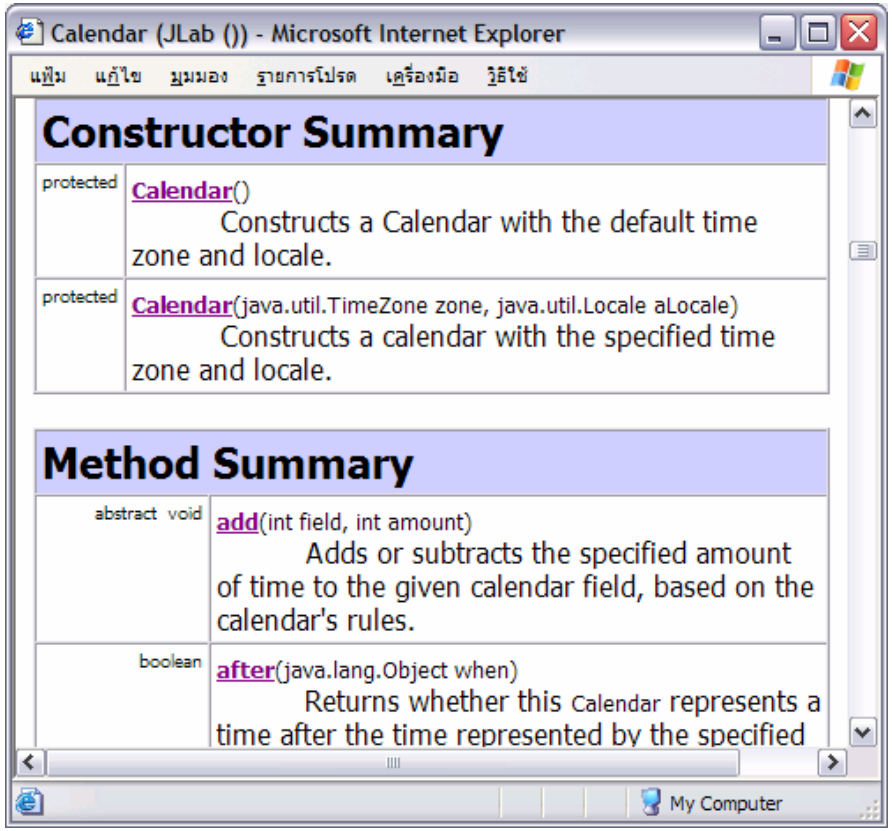
Static factory : สร้างออปเจกต์ของ subtype ได้

```
public class Collections {
  public static Collection unmodifiableCollection(Collection c) {
    return new UnmodifiableCollection(c);
  }
  static class UnmodifiableCollection implements Collection,
                                                 Serializable {
    Collection c;
    UnmodifiableCollection(Collection c) {...}
    public int size() { return c.size(); }
    public boolean isEmpty() { return c.isEmpty(); }
    public Iterator iterator() {...}

    public boolean add(E o) {
      throw new UnsupportedOperationException();
    }
    public boolean remove(Object o) {
      throw new UnsupportedOperationException();
    }
    ...
  }
  ...
}
```



Static factory method : ข้อด้อย

- ไม่สามารถ subclass คลาสที่ไม่มี public และ protected constructors
- javadoc สร้าง API doc
  - constructors : เด่น
  - static factory : ไม่เด่น เหมือนเมท็อดทั่วไป



มักใช้ชื่อ valueOf กับ getInstance เป็น static factory method

Singleton : คลาสที่มีแค่ออปเจกต์เดียว

- class ที่ผลิตออปเจกต์เพียงหนึ่งตัวเท่านั้น
  - ใส่ private ให้กับ constructor
  - มี private static field ที่เก็บ instance เดียวของคลาส
  - มี public static factory method ให้

```
public class LogManager {
  private java.io.PrintStream out;

  private LogManager(PrintStream out) { this.out = out; }

  public void log(String msg) { System.out.println(msg); }
  static private LogManager instance;
  static public LogManager getInstance() {
    if (instance == null) {
      instance = new LogManager(System.out);
    }
    return instance;
  }
}
```

คลาสที่ไม่มีออปเจกต์เลย

- utility class มีแต่
  - static methods กับ static final fields
- แค่ใส่ private no-argument constructor
- อย่าใช้ abstract class เป็นเครื่องมือไม่ให้สร้างออปเจกต์

```
public class Arrays {
  private Arrays() {}  // non-instantiability.

  public static void sort(long[] a) {...}

  public static int binarySearch(long[] a, long key) {...}

  public static void fill(long[] a, long val) {...}
  ...
}
```

อย่าสร้างออปเจกต์ที่มีค่าซ้ำ ๆ

```
String s1 = "JAVA";
String s2 = new String("JAVA");   // DON'T DO THIS
System.out.println(s1 == "JAVA"); // true
System.out.println(s2 == "JAVA"); // false
```

```
public class Util {
  public static boolean isPrimitive(String s) {
    String[] primitives = { "boolean", "byte",
          "char", "double", "float", "int", "long", "short" };
    return Arrays.binarySearch(primitives, s) >= 0;
  }
  ...
```

```
public class Util {
  private static final String[] primitives = { "boolean", "byte",
          "char", "double", "float", "int", "long", "short" };
  public static boolean isPrimitive(String s) {
    return Arrays.binarySearch(primitives, s) >= 0;
  }
  ...
```