Programming Techniques

2301260

วิชานี้

- การเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming)
- ใช้ภาษาจาวา (Java)
- สร้าง Class และ สร้าง object จาก class
- สร้าง subclass โดย inheritance

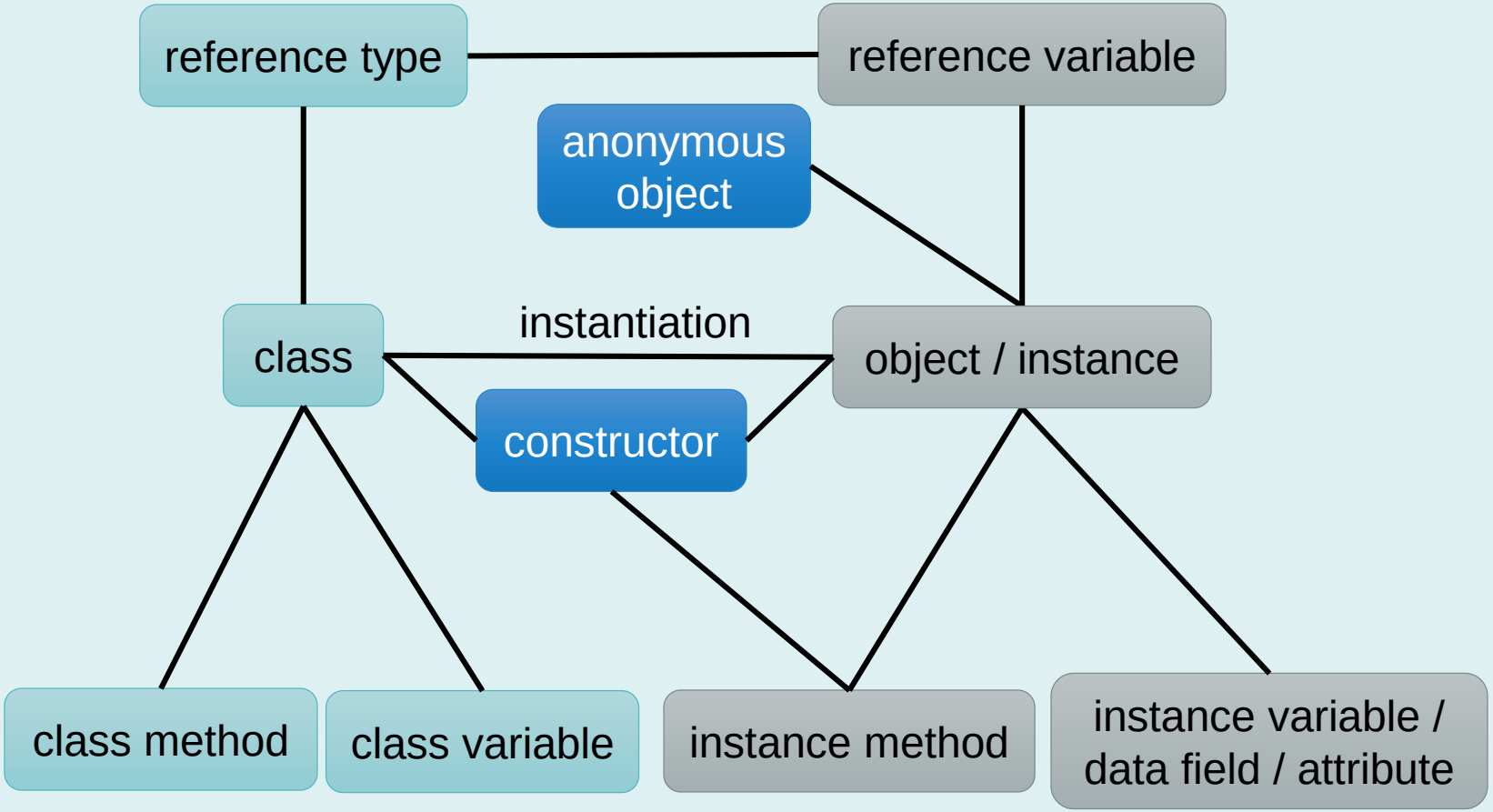
โปรแกรมทำงานกับค่า (values)

- Control flow
  - การทำงานแบบทางเลือก (if structure)
  - การทำงานแบบวนซ้ำ (loops): while loops, for loops
  - การเรียกใช้โปรแกรมย่อย (subprogram, function, method)
- ค่า
  - ค่าคงที่ (constants) ที่เขียนแทนในโปรแกรมด้วย literal เช่น `27, 3.1416, True`
  - ชนิดของข้อมูล (data types)
    - Primitive data types: int, float,…
  - ตัวแปร (variables)

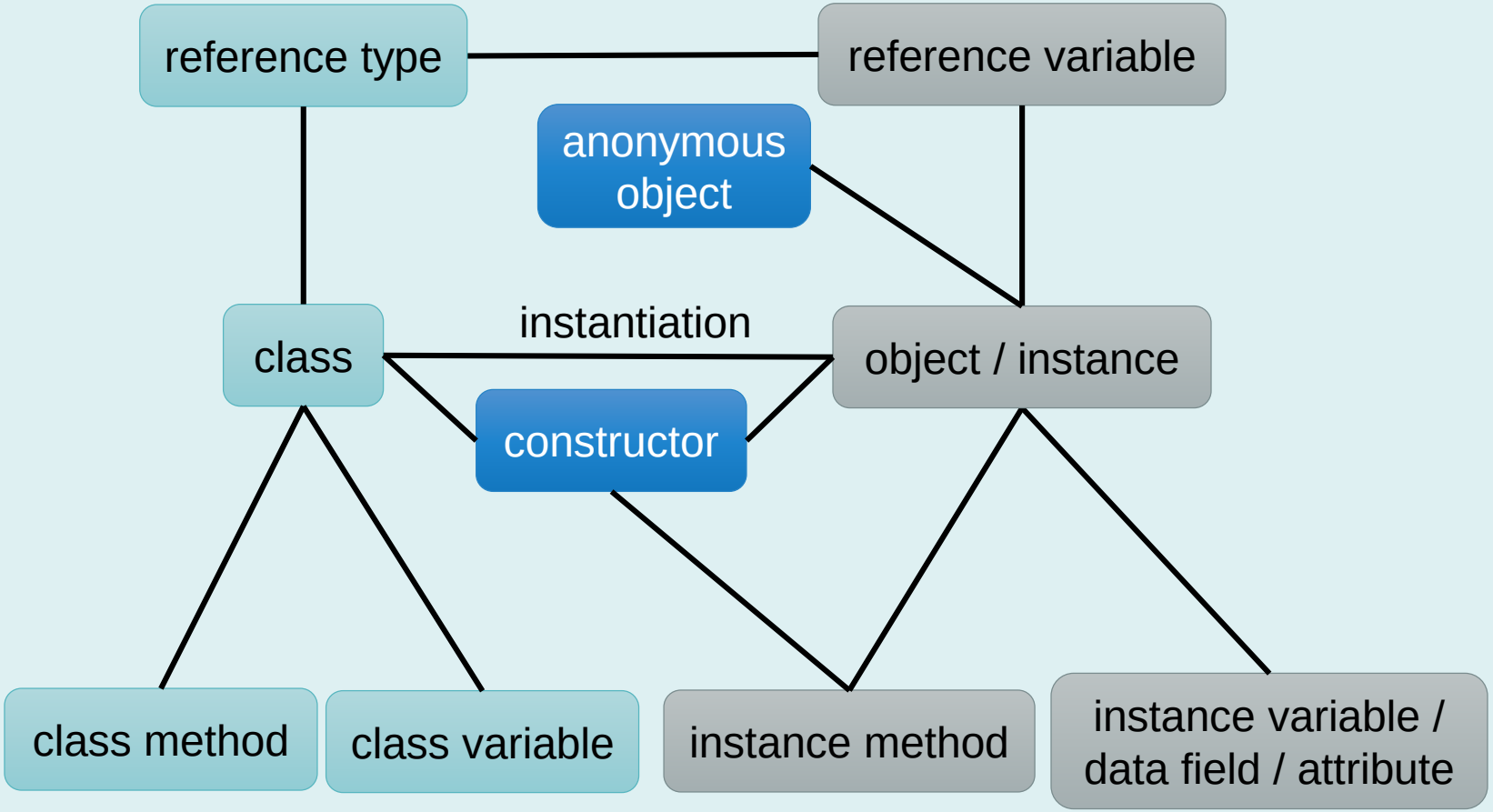
ค่าในโปรแกรมเชิงวัตถุ

- ภาษาโปรแกรมเชิงวัตถุมีวัตถุ (objects) ที่เป็นชนิดข้อมูลทีซับซ้อน (complex data types)
  - มีคลาสที่สร้างไว้แล้ว (Built-in class) เป็นชนิดข้อมูลที่ให้ใช้สร้างวัตถุได้เลย
  - โปรแกรมเมอร์สร้างคลาสที่ต้องการใช้เองได้ (User-defined classes)
- วัตถุมีค่า (value) ที่สร้างมาจากคลาส โดยที่คลาสเป็นชนิดข้อมูล
- วัตถมีพฤติกรรม (behavior) ที่ระบุไว้ด้วยโปรแกรมย่อยที่เรียกว่าเมทอด (method)
- Encapsulation
  - มีตัวกระทำของวัตถุในแต่ละคลาสที่กำหนดไว้ด้วยเมทอดของคลาสนั้น
  - ซ่อนรายละเอียดการทำงานของเมทอด

Objects และ Classes

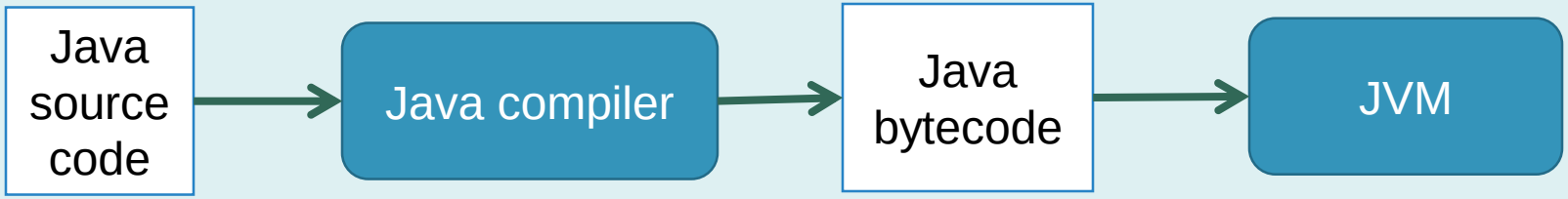


Objects และ Classes



Java

- ภาษาเชิงวัตถุ (Object-oriented language)
- โปรแกรมรันบน Java Virtual Machine (JVM)
- เน้นความถูกต้องของโปรแกรม (safety, reliability)
  - Rigorous type checking
  - Exception handling
- Portability: Write once, run anywhere



Python เทียบกับ Java

- Object-oriented language
- **ไม่ต้อง** ประกาศตัวแปร
- **Dynamic** type
  - ตัวแปรเปลี่ยนชนิดได้

  .
- แต่ละคำสั่งจบในบรรทัด

```
x = 6
```

- กำหนด block ด้วยการย่อหน้า (indentation)

```
if x==1:
   x=x+1
   y=2*x
```

- Object-oriented language
- **ต้อง** ประกาศตัวแปร
- **Static** type
  - ตัวแปรแต่ละตัวไม่เปลี่ยนชนิดตลอดชีวิต (lifetime) ของตัวแปร
- แต่ละคำสั่งจบด้วย ;

```
x = 6;
```

- กำหนด block ด้วย { ... }

```
if (x==1)
{  x=x+1;   y=2*x;
}
```

การประกาศตัวแปรในภาษาจาวา

type varName [= *value*][, *varName* [= *value*] ...] ;

- *type* อาจเป็น int, float, double, boolean, ...

ตัวอย่าง

```
int sum=0;
float intRate, interest;
boolean done=false, primeFlag=true;
```

Comment

```
// this is one-line comment


/* this is a multiple-line comment
   which can go over many lines
   until the closing star (*) slash (/)
   is found.
*/
```

Operations

- Arithmetic operators: `+` `-` `*` `/` `%`
- Bitwise operators: `~` `&` `|` `^` `>>` `<<` `>>>`
- Increment / decrement:
  - `++` `--` (prefix) : `z = ++x * y` means `x=x+1; z = x *y;`
  - `++` `--` (postfix) : `z = x++ * y` means `z = x *y; x=x+1;`
- Relational operators: `>` `>=` `<` `<=` `==` `!=`
- Logical operators: `&&` `||` `!`
- Assignments:
  - `=`
  - `+= -= *= /= %= ~= &= |= ^= >>= <<= >>>=`
  - `x += y;` means `x = x+y;`

Precedence

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >=
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

โครงสร้าง if ในภาษาไพธอนและจาวา

Python

```
if x<1:
   y='A'
elif x<2:
   y='B'
else:
   y='C'
```

Java

```
if (x<1)
y='A';
else if (x<2)
y= 'B';
else y='C';
```

โครงสร้าง if ในภาษาไพธอนและจาวา

Python

```
if x<1:
    y='A'
    x=0
elif x<2:
    y='B'
    x=10
else:
    y='C'
    x=100
```

Java

```
if (x<1) {
  y='A'; x=0;
}
else if (x<2) {
  y= 'B'; x=10;
}
else {
  y='C';
  x=100;
}
```

โครงสร้าง while ในภาษาไพธอนและจาวา

Python

```
while x<1:
    y='A'
    x = x+1
```

Java

```
while (x<1) {
    y ='A';
    x =x+1;
}
```

โครงสร้าง for ในภาษาไพธอนและจาวา

Python

```
for i in range(1,9,2):
    t=t+i
    s=s*i
```

Java

```
for (int i=1;i<9;i=i+2){
    t=t+i;
    s=s*i;
}
```

Error

Compile-time error

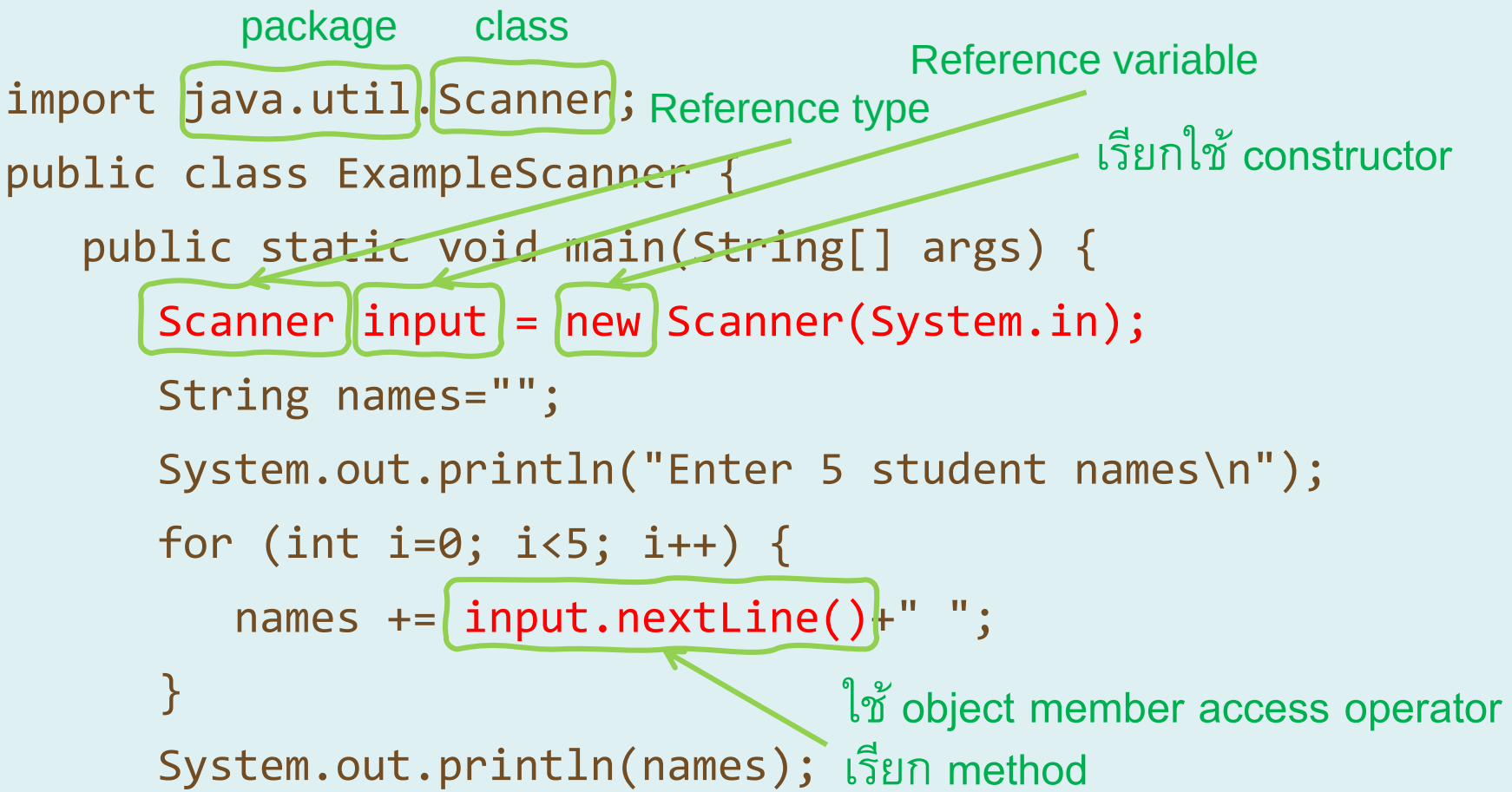
- Syntax error
- Type error

Run-time error

- Logical error : ผลลัพธ์ไม่ถูกต้อง
- บางครั้งขึ้นกับ input : ชนิดของข้อมูล

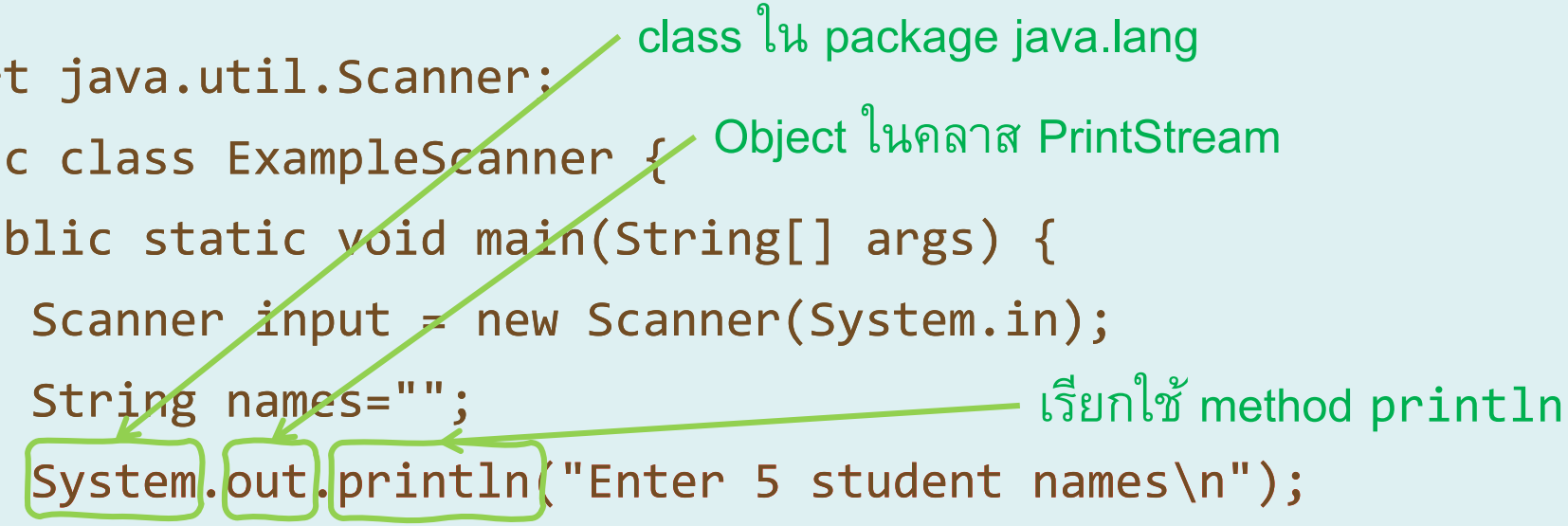
ตัวอย่างการใช้ built-in class ในภาษาจาวา

การใช้คลาส Scanner



การใช้คลาส PrintStream

```
import java.util.Scanner;
public class ExampleScanner {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String names="";
        System.out.println("Enter 5 student names\n");
        for (int i=0; i<5; i++) {
            names += input.nextLine()+" ";
        }
        System.out.println(names);
    }
}
```



Class Hierarchy

- java.lang.**Object**
  - java.io.**Console** (implements java.io.Flushable)
  - java.io.**File** (implements java.lang.Comparable<T>, java.io.Serializable)
  - java.io.**FileDescriptor**
  - java.io.**InputStream** (implements java.io.Closeable)
    - java.io.**ByteArrayInputStream**
    - java.io.**FileInputStream**
    - java.io.**FilterInputStream**
      - java.io.**BufferedInputStream**
      - java.io.**DataInputStream** (implements java.io.DataInput)
      - java.io.**LineNumberInputStream**
      - java.io.**PushbackInputStream**
    - java.io.**ObjectInputStream** (implements java.io.ObjectInput, java.io.ObjectStreamConstants)
    - java.io.**PipedInputStream**
    - java.io.**SequenceInputStream**
    - java.io.**StringBufferInputStream**
  - java.io.**ObjectInputStream.GetField**
  - java.io.**ObjectOutputStream.PutField**
  - java.io.**ObjectStreamClass** (implements java.io.Serializable)
  - java.io.**ObjectStreamField** (implements java.lang.Comparable<T>)
  - java.io.**OutputStream** (implements java.io.Closeable, java.io.Flushable)
    - java.io.**ByteArrayOutputStream**
    - java.io.**FileOutputStream**
    - java.io.**FilterOutputStream**
      - java.io.**BufferedOutputStream**
      - java.io.**DataOutputStream** (implements java.io.DataOutput)
      - java.io.**PrintStream** (implements java.lang.Appendable, java.io.Closeable)
    - java.io.**ObjectOutputStream** (implements java.io.ObjectOutput, java.io.ObjectStreamConstants)
    - java.io.**PipedOutputStream**
  - java.security.**Permission** (implements java.security.Guard, java.io.Serializable)
    - java.security.**BasicPermission** (implements java.io.Serializable)
      - java.io.**SerializablePermission**
    - java.io.**FilePermission** (implements java.io.Serializable)

การใช้ println ในคลาส PrintStream

println

`public void println(int x)`

Prints an integer and then terminate the line. This method behaves as though it invokes `print(int)` and then `println()`.

Parameters:

`x - The int to be printed.`

println

`public void println(char[] x)`

Prints an array of characters and then terminate the line. This method behaves as though it invokes `print(char[])` and then `println()`.

Parameters:

`x - an array of chars to print.`

println

`public void println(String x)`

Prints a String and then terminate the line. This method behaves as though it invokes `print(String)` and then `println()`.

Parameters:

`x - The String to be printed.`

println

`public void println(Object x)`

Prints an Object and then terminate the line. This method calls at first String.valueOf(x) to get the printed object's string value, then behaves as though it invokes `print(String)` and then `println()`.

Parameters:

`x - The Object to be printed.`

overload

```
/* See
https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html
https://docs.oracle.com/javase/8/docs/api/java/time/Period.html
 */
import java.time.LocalDate; // import the LocalDate class
import java.time.Period;    // import the Period class
public class ExampleDate {
public static void main(String[] args) {
        int days, months;
        LocalDate today = LocalDate.now();
        LocalDate tomorrow = today.plusDays(1);

        LocalDate startDate = LocalDate.parse("2020-11-20");
        Period duration = Period.between(startDate, today);
        days = duration.getDays();
        months = duration.getMonths();
        System.out.println(today);
        System.out.println(startDate);
        System.out.println("days: "+days+", months: "+ months);
    }
}
```

```
/* See
https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html
https://docs.oracle.com/javase/8/docs/api/java/time/Period.html
 */
import java.time.LocalDate; // import the LocalDate class
import java.time.Period;    // import the Period class
public class ExampleDate {
public static void main(String[] args) {
        int days, months;
        LocalDate today = LocalDate.now();
        LocalDate tomorrow = today.plusDays(1);

        LocalDate startDate = LocalDate.parse("2020-11-20");
        Period duration = Period.between(startDate, today);
        days = duration.getDays();
        months = duration.getMonths();
        System.out.println(today);
        System.out.println(startDate);
        System.out.println("days: "+days+", months: "+ months);
    }
}
```

OUTPUT:

```
2021-01-06
2020-11-20
days: 17, months: 1
```

การใช้คลาส LocalDate , Period

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP — Java™ Platform Standard Ed. 8

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES ALL CLASSES

Package java.time

The main API for dates, times, instants, and durations.

See: Description

Class Summary

Class	Description
Clock	A clock providing access to the current instant, date and time using a time-zone.
Duration	A time-based amount of time, such as '34.5 seconds'.
Instant	An instantaneous point on the time-line.
LocalDate	A date without a time-zone in the ISO-8601 calendar system, such as 2007-12-03.
LocalDateTime	A date-time without a time-zone in the ISO-8601 calendar system, such as 2007-12-03T10:15:30.
LocalTime	A time without a time-zone in the ISO-8601 calendar system, such as 10:15:30.
MonthDay	A month-day in the ISO-8601 calendar system, such as --12-03.
OffsetDateTime	A date-time with an offset from UTC/Greenwich in the ISO-8601 calendar system, such as 2007-12-03T10:15:30+01:00.
OffsetTime	A time with an offset from UTC/Greenwich in the ISO-8601 calendar system, such as 10:15:30+01:00.
Period	A date-based amount of time in the ISO-8601 calendar system, such as '2 years, 3 months and 4 days'.
Year	A year in the ISO-8601 calendar system, such as 2007.
YearMonth	A year-month in the ISO-8601 calendar system, such as 2007-12.
ZonedDateTime	A date-time with a time-zone in the ISO-8601 calendar system, such as 2007-12-03T10:15:30+01:00 Europe/Paris.
ZoneId	A time-zone ID, such as Europe/Paris.
ZoneOffset	A time-zone offset from Greenwich/UTC, such as +02:00.

การใช้คลาส Random

```
import java.util.Random;

public class Example {
  public static void main(String[] args) {

    //create random number generator
    Random random = new Random();
    //returns pseudorandom integer 0-5
    System.out.println("Random int 0-5:"+random.nextInt(6));
    // Returns the next pseudorandom double
    System.out.println("Random double : "+ random.nextDouble());
  }
}
```

การใช้คลาส Calendar

```
import java.util.GregorianCalendar;
import java.util.Calendar;
public class ExampleDate {
  public static void main(String[] args) {
  // Creating a GregorianCalendar obj to get current date time
  GregorianCalendar gcal = new GregorianCalendar();
  System.out.println("Date: "+
    gcal.get(Calendar.MONTH) + " "+ gcal.get(Calendar.DATE) + ", " +
    gcal.get(Calendar.YEAR) + "\n"+ "Time: " +
    gcal.get(Calendar.HOUR) + ":"+ gcal.get(Calendar.MINUTE) + ":" +
    gcal.get(Calendar.SECOND) +" "+ gcal.get(Calendar.AM_PM) + "\n" +
    "Time Zone: " + gcal.getTimeZone().getDisplayName() );
}
```

Output

```
Date: 0 15, 2021
Time: 11:48:12 0
Time Zone: Indochina Time
```

การใช้คลาส Calendar

```
import java.util.GregorianCalendar;
import java.util.Calendar;
public class ExampleDate {
  public static void main(String[] args) {
  // create an object for 5 Feb 2000
GregorianCalendar bd=new GregorianCalendar(2000,Calendar.FEBRUARY,5);
  System.out.println("Date: "+ bd.get(Calendar.MONTH) + " "+
     bd.get(Calendar.DATE) + ", " + bd.get(Calendar.YEAR));
  bd.add(Calendar.DATE, 24);
  System.out.println("Date: "+ bd.get(Calendar.MONTH) + " "+
     bd.get(Calendar.DATE) + ", " + bd.get(Calendar.YEAR));    }
```

Output

```
Date: 1 5, 2000
Date: 1 29, 2000
```

การใช้คลาส Calendar

```
import java.util.GregorianCalendar;
import java.util.Calendar;
public class ExampleDate {
  public static void main(String[] args) {
  // create an object for 5 Feb 2000
GregorianCalendar bd=new GregorianCalendar(2000,Calendar.FEBRUARY,5);
  System.out.println("Date: "+ bd.get(Calendar.MONTH) + " "+
     bd.get(Calendar.DATE) + ", " + bd.get(Calendar.YEAR));
  bd.add(Calendar.DATE, 25);
  System.out.println("Date: "+ bd.get(Calendar.MONTH) + " "+
     bd.get(Calendar.DATE) + ", " + bd.get(Calendar.YEAR));    }
```

Output

```
Date: 1 5, 2000
Date: 2 1, 2000
```