

DES103 LAB02:

Class components in more details, the `this` keyword, instance

Learning Objectives

- To study classes with more details of properties, constructors and methods
- To study classes with constructors with arguments
- To study classes with no constructor
- To study the use of the `this` keyword
- To study the instance creation
- To study classes with methods of the same name

1.1 Class's Properties

Class's Properties are sometimes called *attributes*, *fields*, or *variables*

Types of class's properties

primitive type: `int`, `float`, `double`, `char`, ...

object type: type created from a class

Type	Default value
<code>int</code>	<code>0</code>
<code>double</code> , <code>float</code>	<code>0.00</code>
<code>char</code>	<code>'u0000'</code>
<code>boolean</code>	<code>false</code>
reference type (object type)	<code>null</code>

1.2 Constructors

- Constructors are used to construct an instance of an object with user-defined properties
- It must have exactly the same name as the class name
- The constructors must have no return type.
- A class can have no constructor at all.
- A class can have multiple constructors but they must not have the same list of argument types

1.3 Types of Constructors

1. *No argument constructor*: constructor without argument. The initial values of the properties will be used. If the properties are not initially set, the default values below are used.
2. *Argument constructor*: constructor with argument

No-show constructor: is provided by JRE when the class has no constructor. This type of constructor has no argument and has empty content.

1.4 Class's Methods

- Methods are program functions that define the behaviors of the object.
- A class can contain no method or multiple methods.
- Methods can be overloaded. i.e. they are allowed to have the same name but different parameter lists.

Ex.

```
int    B(int x, String y) {...}  
void   B(String y) {...}
```

The Java compiler determines which method is used based on the method signature.

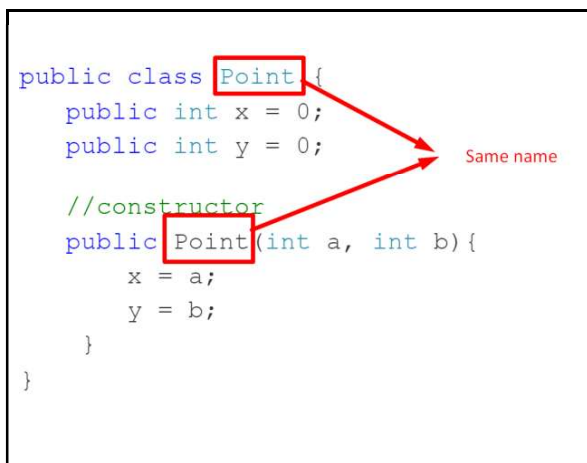
1.5 The *this* keyword

- The *this* keyword is used for differentiating the arguments and the properties variables when they have the same name
- Referring to another constructor.
- Remark that for this case the call of this must appear before any other statements in the constructor.

Note that: Within an instance method or a constructor, this is a reference to the *current object* — the object whose method or constructor is being called. You can refer to any member of the current object from within an instance method or a constructor by using this.

The most common reason for using the *this* keyword is because a field is shadowed by a method or constructor parameter.

For example, the `Point` class was written like this:



```
public class Point {  
    public int x = 0;  
    public int y = 0;  
  
    //constructor  
    public Point(int a, int b) {  
        x = a;  
        y = b;  
    }  
}
```

but it could have been written like this:

```
public class Point {  
    public int x = 0;  
    public int y = 0;  
  
    //constructor  
    public Point(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```

Reference :: <https://docs.oracle.com/javase/tutorial/java/javaOO/thiskey.htm>

LAB02 Exercises

Students should follow lab instructions and regulations.

Ask your TA to check your finished exercises and attach them to Google Class.

Be noticed that for all lab exercises, you need to define your *Java* project as the following name format:

<Student ID>_<Lab number>_<Exercise name>

If your student's ID is 6122300300, the name format of your java project should be:

6122300300_LAB02_PeopleID



Exercise 1: (2 points)

Project Name: <StudentID>_LAB02_PeopleID

Create a Java project <Student ID>_Lab2_Exercise1. Write a java class `Date`. The class `Date` has three properties: `int day`, `int month`, `int year`. It has one constructor that takes 3 arguments: `int day`, `int month`, `int year`. The constructor assigns the input arguments to its corresponding properties. The class `Date` has only one method: `void printDate()` which prints the date in the format of day-month-year.

Hint: Using this with a field/attribute



Exercise 2: (2 points)

Project Name: <StudentID>_LAB02_PeopleID

Learning objective: To learn how to define class, attribute, constructor, and method in class by Java programming.

Write a java class `Name`. The class `Name` has two properties: `String firstName` and `String lastName`. It has one constructor that takes 2 arguments: `String firstName`, `String lastName`. The constructor assigns the input arguments to its corresponding properties. The class `Name` has only one method: `void printName()` which prints the `firstName` `lastName` with a space between them, for example, James Mars



Exercise 3: (2 points)

Project Name: <StudentID>_LAB02_PeopleID

Learning objective: To learn how to define multiple constructors in Java programming.

Write a java class `Address`. The class `Address` has 7 properties: `String houseNo`, `String soi`, `String road`, `String subDistrict`, `String district`, `String province`, and `String postcode`.

These properties are initialized to a `String` "-".

It has two constructors. The first one that takes 7 arguments: `String houseNo`, `String Soi`, `String Road`, `String subDistrict`, `String district`, `String province`, and `String postcode`. The 7-argument constructor assigns the input arguments to its corresponding properties. The second constructor takes only two arguments: `String province`, and `String postcode`. It assigns the input arguments to their corresponding properties.

The class `Address` has two methods:

- The first method is `void printFullAddress()` which prints the address in the following format accordingly,
 - *houseNo, road, subDistrict, district, province, postcode*
 - **Example** 81/9, ChiangMai-HangDong, Sunpakwan, Hang Dong, Chiang Mai, 50230
- The second method is `void printShortAddress()` which prints the address in the following format accordingly,
 - *district, province*
 - **Example** Hang Dong, Chiang Mai



Exercise 4 (2 points)

Project Name: <StudentID>_LAB02_PeopleID

Learning objective: To learn how to define multiple constructors, create an object, and use the object in Java programming.

Write a class `PeopleID`. The `PeopleID` class has four properties: `Name name`, `String ID`, `Date dateOfBirth`, and `Address address`.

It has 2 constructors: A 2-argument constructor and a 4-argument constructor. The 2-argument constructor takes `Name name`, and `String ID` as inputs. It sets the properties `name` and `ID` as the input argument. The 4-argument constructor takes `Name name`, `String iD`, `Date dateOfBirth`, and `address` as inputs. It uses the `this` keyword to call a 2-argument constructor that takes `name` and `ID` as an input for initializing the properties `name` and `ID`, and also initializes the `dateOfBirth` and `address` to their corresponding properties.

It has only 1 method: `void printPeopleID()` which calls `printName()` of the `name` property, prints out the `ID` property of this class, calls `printDate()` of the `date` property. and calls `printFullAddress()` of the `address` property.



Exercise 5 (2 points)

Project Name: <StudentID>_LAB02_PeopleID

Learning objective: To learn how to define multiple constructors, create an object, and use the object in Java programming.

Write a class `TestPeopleID`. Then add the statements according to the instructions provided in the comments (`/*.....*/`).

Student can copy the class `TestPeopleID` skeleton as follows:

```
public class TestPeopleID {
    public static void main(String [] args){
        /*use the keyword new to construct a Date object with day=23 month=4
        and year=2000, assign this Date object to a variable dobObj of type
        Date */

        /*use the keyword new to construct a Name object with firstName =
        "Wiangping" and lastName = "Sangjan", assign this Name object to a
        variable nameObj of type Name*/

        /*use the keyword new to construct an Address object with the
        following information houseNo="81/9", Soi= "2", road = "ChiangMai-
        HangDong", subDistrict="Sunpakwan", district= "Hang Dong", province=
        "Chiang Mai", and postcode= "50230", assign this Address object
        to a variable addressObj of type Address */

        /* assign "3-5015-00274-987" to a variable idObj of type String */

        /*use the keyword new to construct a PeopleID from the dobObj,
        nameObj, idObj and addressObj you just constructed, assign this
        peopleID object to a variable peopleIDObj of type PeopleID */

        /*Call the method printPeopleID() of peopleIDObj */

        System.out.println("-----");

        System.out.print("The name of peopleIDObj is ");
        /*printout the name of peopleIDObj by calling the method printName()
        of the name property of peopleIDObj */

        System.out.print("The postcode of the peopleIDObj is ");
        /*use System.println to print the postcode of the address property
        of peopleIDObj */

        System.out.println("-----");

    }
}
```

Running output should be as below:

```
Wiangping Sangjan
3-5015-00274-987
23-4-2000
81/9, , ChiangMai-HangDong, Sunpakwan, Hang Dong, Chiang Mai, 50230
-----
The name of peopleIDObj is Wiangping Sangjan
The postcode of the peopleIDObj is 50230
-----
```