

# [DES103(Y23S2)–LAB02–OVERVIEW]

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

---

## Learning Objectives

1. Detailed Study of Class Properties, Constructors, and Methods
2. Study of Classes with Constructors and Arguments
3. Exploration of Classes with No Constructor
4. Understanding the Use of the "*this*" Keyword
5. In-Depth Look at Instance Creation
6. Study of Classes with Methods of the Same Name

## 1.1 Class's Properties

- ❖ Also known as *attributes*, *fields*, or *variables* within a class.
- ❖ Represent characteristics or data associated with an **object**.
  - Primitive type properties: default values for primitive type properties

```
public class Example {  
    int intValue;           // Default value: 0  
    double doubleValue;    // Default value: 0.00  
    float floatValue;      // Default value: 0.00  
    char charValue;        // Default value: '\u0000'  
}
```

- Object type properties: type created from a class.  
Default value for reference type (object type) properties

```
public class Example {  
    String stringValue;     // Default value: null  
    MyClass customObject;  // Default value: null  
}
```

## 1.2 Constructors

- ❖ **Constructors** are special methods used to initialize and construct an instance of an object.
- ❖ They are called when an **object** is created from a class.
- ❖ **Purpose of Constructors**
  - Constructors initialize the properties or attributes of an object with user-defined values.
  - Ensure that an object is in a valid state upon creation.
- ❖ **Naming Convention:**  
Constructors must have the exact same name as the class they belong to.

```
public class MyClass {  
    public MyClass() {  
        // Constructor code here  
    }  
}
```

- ❖ **Return Type:**
  - Constructors have no return type.
  - Their purpose is to initialize, not to return a value.

❖ **No Constructor in a Class**

- It's possible for a class to have no constructor at all.
- In such cases, a default constructor is implicitly provided.

❖ **Multiple Constructors**

- A class can have multiple constructors.
- Constructors must have different lists of argument types.

```
public class Example {  
    public Example() {  
        // Default constructor  
    }  
  
    public Example(int value) {  
        // Constructor with an int argument  
    }  
    // Additional constructors with different argument  
    types  
}
```

- ❖ Constructors are essential for initializing objects.
- ❖ They ensure that objects are created in a valid and expected state.
- ❖ Follow naming conventions and rules for defining constructors.

### 1.3 Types of Constructors

1. *No Argument Constructor*: Constructor without any arguments. The initial values of the properties will be used. If the properties are not initially set, the default values below are used.

```
public class Example {  
    public Example() {  
        // No argument constructor  
    }  
}
```

2. *Argument Constructor*: Constructor with one or more arguments. Accepts parameters to initialize properties based on provided values.

```
public class Example {  
    public Example(int value, double anotherValue) {  
        // Argument constructor  
        // Initialize properties using provided values  
    }  
}
```

3. *No-Show Constructor (Default Constructor)* provided by *Java Runtime Environment (JRE)* when a class has no explicit constructor.

```
public class Example {  
    // No explicit constructor  
    // Default constructor is provided by JRE  
}
```

## 1.4 Class's Methods

- ❖ Methods are program functions that define the behaviors of an *object*. They encapsulate the functionality associated with the object's actions.
- ❖ A class can contain no method or multiple methods.
- ❖ Methods contribute to the behavior and functionality of the class.
- ❖ Overloading Methods
  - Methods can be overloaded, meaning they are allowed to have the same name but different parameter lists.
  - Overloaded methods provide flexibility and convenience.

```
public class Example {  
    // Overloaded methods  
    int B(int x, String y) {  
        // Method implementation  
    }  
    void B(String y) {  
        // Method implementation with different  
        parameters  
    }  
}
```

- Example Method Overloading

```
public class MathOperations {  
    // Overloaded methods  
    int add(int a, int b) {  
        // Method implementation  
    }  
  
    double add(double a, double b) {  
        // Overloaded method with different  
        parameter types  
    }  
}
```

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

## 1.5 The “this” keyword

- ❖ The “*this*” keyword is used to differentiate between class properties and method parameters when they share the same name.
- ❖ Differentiating Variables  
When properties and parameters have the same name, this helps clarify which variable is being referred to.

```
public class Example {  
    private int value;  
  
    public void setValue(int value) {  
        // Using "this" to differentiate between property  
        and parameter  
        this.value = value;  
    }  
}
```

```
}
```

❖ Referring to Another Constructor

- “*this*” is also used to invoke another constructor within the same class.
- The call to “*this*” must appear before any other statements in the constructor.

```
public class Example {  
    private int value;  
  
    // Constructor with an argument  
    public Example(int value) {  
        this.value = value;  
    }  
  
    // Another constructor invoking the first one  
    public Example() {  
        this(0); // Calling the constructor with an argument  
        // Other statements can follow after the constructor call  
    }  
}
```

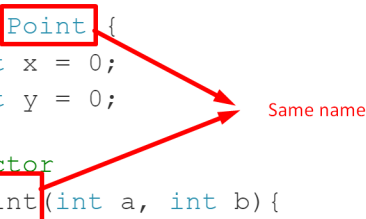
❖ Benefits of Using “*this*”

- Enhances code clarity by explicitly specifying the scope of variables.
- Facilitates constructor overloading within the same class.

**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;  
    }  
}
```



Same name

- but it could have been written like this:

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

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

## [DES103(Y23S2)–LAB02–EXERCISES]

Students **MUST** adhere to the lab instructions and regulations provided below. Please consult your TA to review your completed exercises and submit them on 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:

6422300208\_LAB02\_PeopleID



### Exercise 1: (2 points)

- **Project Name:** <StudentID>\_LAB02\_PeopleID
- **Instruction:**
  1. Create a Java project <Student ID>\_Lab2\_PeopleID.
  2. Write a java class Date.
  3. The class Date has three properties: `int day`, `int month`, `int year`.
  4. It has one constructor that takes 3 arguments: `int day`, `int month`, `int year`.
  5. 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 define class, attribute, constructor, and method in class by Java programming.
- **Instruction:**
  1. Create a java class, Name.
  2. The Name class has two properties: `String firstName` and `String lastName`.
  3. The Name class has one constructor that takes 2 arguments: `String firstName`, `String lastName`.
  4. The constructor assigns the input arguments to its corresponding properties.
  5. The Name class 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.
- **Instruction:**
  1. Write a java class Address.
  2. The class Address has 7 properties: `String houseNo`, `String soi`, `String road`, `String subDistrict`, `String district`, `String province`, and `String postcode`.
  3. These properties are initialized to a String “-”.

4. 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.
5. 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.
- **Instruction:**
  1. Write a class `PeopleID`. The `PeopleID` class has four properties: `Name name`, `String ID`, `Date dateOfBirth`, and `Address address`.
  2. 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 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.
  3. 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.
- **Instruction:**
  1. Write a class `TestPeopleID`. Then add the statements according to the instructions provided in the comments (`/* ..... */`).

**2. 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 =
        "Somchai" and lastName = "Yodying", 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:**

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