

# WIX1002

## Fundamentals of Programming

---

### Chapter 8

### Class





# Contents

- Introduction
- Constructor
- Instance Variables
- Static Variable
- Class method
- Encapsulation
- Method Overloading
- Wrapper Class
- Package



# Introduction

- Java is an Object-oriented programming language (OOP).
- OOP is a programming methodology that views a program as similarly consisting of **objects** that interact with each other by means of **actions**.
- An object is a software bundle of related **state and behavior**. Software objects are often used to model the real-world objects in everyday life.
- Objects of the same kind are in the same **class**. Example, a Perodua Myvi white car and a Perodua Myvi blue car are belongs to the same class.
- The actions of an object are called **methods**.



# Introduction

- Java consists of objects from various classes interacting with one another.
- A class is a prototype from which objects are created. It models the state and behavior of a real-world object.
- A class consists of **constructor**, data items (**instance variables**) and **methods**.

- To define a class

```
public class className {  
    // instance variables  
    // constructors  
    // methods  
}
```



# Introduction

- Once the class has been defined. The object of the class can be created in the tester program.
- To create the object of the class  
className obj = **new** className();

```
public class Test {  
    public static void main(String[] args) {  
        person p = new person();  
        // ...  
    }  
}
```



# Constructor

- Constructor is a special variety of method that used to initialize instance variable.
- Constructor must have the **same name as the class**.
- To define constructor

```
public className() {  
  
}  
public className(parameterType parameterName, ...) {  
  
}
```



# Constructor

- Constructor without parameters is called no-argument constructor.
- Java will automatically creates a no-argument constructor if the class definition does not have a constructor.



# Constructor

```
public person() {  
    name = null;  
    age = 0;  
    gender='\0';  
}  
person p = new person();
```

```
public person(String n, int a, char g) {  
    name = n;  
    age = a;  
    gender=g;  
}  
person p = new person("John", 30, 'M');
```





# Constructor

- A copy constructor is a constructor with one parameter of the same type as the class. It creates an object that is separate, independent object with the instance variables are **exact copy of an argument object**.

```
public person(person p) {  
    if (p==null)  
        System.exit(0);  
  
    name = p.getName();  
    age = p.getAge();  
    gender= p.getGender();  
}
```



# Instance Variables

- An object used instance variables to store its state.
- To declare instance variables  
**accessSpecifier variableType variableName;**  
private String name;  
private int age;  
private char gender;
- The accessSpecifier for instance variable is always private.
- Access specifiers consists of **private, protected and public.**



# Instance Variables

- **public** means that there are no restrictions on where the instance variable can be used.
- **private** means that the instance variable is accessible only by methods of the same class.
- **protected** means that the instance variable is accessible by methods of the same class, subclass or class in the same package.
- A **package** is a namespace for organizing classes and interfaces in a **logical manner**. It makes large software projects easier to manage.



# Instance Variables

Specifier	class	subclass	package	world
private	Y			
protected	Y	Y	Y	
public	Y	Y	Y	Y



# Instance Variables

- In Java method definition, the **this** keyword can be used as the **calling object**.
- The **this** parameter refer to the **reference to the current object**, the object whose method or constructor is being called.

```
public void setName(String name) {  
    this.name = name;  
}
```



# Static variable

- A static variable is a variable that belongs to the class. There is only **one copies of the static variable for all the objects**.
- A static variable can be used by **objects to communicate** between the objects.
- To declare a static variable
  - `private static int counter;`
- To declare a static constants
  - `private static final int MAX = 100;`
- **A static method can access the static variables but can't access the instance variables.**



# Class Method

- **Accessor method** are used to **get the value** of instances variable

```
public String getName() {  
    return name;  
}
```

- **Mutator method** are used to **set or modify** the instances variable.

```
public void setName(String n) {  
    name = n;  
}
```



# Class Method

```
public class Test {  
    public static void main(String[] args) {  
        person p = new person();  
        p.setName("Alvin");  
        System.out.println("The name is " + p.getName());  
    }  
}
```





# Class Method

- In Java, the class definition usually contains the **equals method** and the **toString method**.
- An equals method compare the calling object to another object and return true if both objects are equal.

```
public boolean equals(Person p) {  
    if (name.equals(p.getName()))  
        return true;  
    else  
        return false;  
}
```



# Class Method

- The toString method return a string representation of the data in the calling object.

```
public String toString() {  
    return "Name is " + this.name;  
}
```



# Encapsulation

- **Encapsulation** means that the data and actions are combined into a single item which is a Class.
- It is used to group software into a unit that is easy to use with the simple interface.
- By defining the **Application Programming Interface (API)**, programmer can use it for any implementation.



# Method Overloading

- **Method overloading** allows one class to have more than one definitions of a single method name.
- When a method is overloading, the definition must have different signatures either in **different number of parameters** or **some parameter position must be of differing types**.
- Java does not support **operator overloading**.



# Method Overloading

```
public void setValue(String n) {  
    name = n;  
}  
public void setValue(String n, int a) {  
    name = n;  
    age = a;  
}  
public void setValue(String n, int a, char g) {  
    name = n;  
    age = a;  
    gender = g  
}
```



# Wrapper Class

- Every Java primitive type has a corresponding wrapper class.
- The wrapper class contains useful predefined constant and static methods.
- Create object of a wrapper class  
`Integer num = new Integer(100);`  
`Integer num1 = 50;`  
`Double price = 10.59;`  
`Character grade = 'C';`



# Wrapper Class

- Each wrapper class contains the instance variable for maximum and minimum value  
Integer.MAX\_VALUE  
Integer.MIN\_VALUE
- The wrapper class also contains the method to convert String value to their type or from their type to String value.
- Integer.parseInt(String) // convert String to integer
- Double.parseDouble(String) // convert String to double
- Integer.toString(intValue) // convert integer to String



# Wrapper Class

- Character Wrapper Class
  - `.toUpperCase(charVariable);`
  - `.toLowerCase(charVariable);`
  - `.isUpperCase(charVariable);`
  - `.isLowerCase(charVariable);`
  - `.isWhitespace(charVariable);`
  - `.isLetter(charVariable);`
  - `.isDigit(charVariable);`
  - `.isLetterOrDigit(charVariable);`





# Package

- A package is a collection of classes that have been grouped together into a directory.
- Each package is given a package name. The package name is the path name to the directory containing the classes in the package.
- In Java, the CLASSPATH variable is used to locate Java packages. The new package will store in the directory **CLASSPATH/packageName** folder.
- Each file in the same package must contain the same package name.
  - `package packageName;`



# Package

- Whenever a package is import, it does not import the subdirectory packages.
- All the classes in the current directory are called default package. The current directory of the default package is the CLASSPATH
- When compiling a Java file, Java use the default CLASSPATH. In order to use different CLASSPATH for compiling a Java file
  - `javac -classpath .;newClassPathDirectory fileName.java`
  - **. refer to current directory**

