Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.ajj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

# [DES103(Y23S2)-LAB04-REVIEW]

## Polymorphism, Abstract, and Interface

### Learning Objectives

1. To learn the concept of Polymorphism.
2. To learn access properties and methods of a variable which is declared as a superclass type but refer to a subclass object.
3. To learn classes that are abstract.
4. To learn methods that are abstract.
5. To learn Interface.
6. To learn the UML of the abstract classes and interface.

### 4.1 Polymorphism

- *Polymorphism* is the concept that one instance can be referred to as the superclass's instance.
- Students can convert an object of one class type to another within an inheritance hierarchy.
- To explain your TA, Student should try to understand the following **example**:

```
Circle circle01 = new Cylinder(0, 0, 1.0, 2.0);
```

This example can say that this variable name `circle01` is declared as a half-blood variable of `Cylinder` class, but the `circle01` is assigned in a polymorphous type as `Circle` class.

### 4.2 Implicit Casting

- An object in Java can be created with a subclass type and be referenced as a superclass type.
- To explain your TA, students SHOULD try to understand the following **example**:

```
String studentAge_String = "12";
int StudentAge = Integer.parseInt(studentAge_String);

float studentScore_Flaot = 19.98f;
int studentScore_Integer = (int) studentScore_Flaot;
```

### 4.3 `instanceof` Operator

- Because a superclass-type variable can come from many subclass-type creations.
- Students can use the operator `instanceof` to check its subclass-type.
- The `instanceof` operator takes two operands:
  - `frontOperand` and `backOperand`

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

- It returns true if the `frontOperand` is an instance of `backOperand`, and returns false otherwise.

- In the superclass can be accessed directly by:
  `super.method` and `super.property`.

## 4.4 Abstract Classes

- Sometimes a superclass is so abstract that it cannot have any specific instances. Such a class is referred to as an abstract class.
- *Abstract class* is used when we want to force all the sub-classes of the abstract class to have specific methods in which the content depends mainly on the subclass.
- Students SHOULD check the following properties to understand the abstract class in Java code
  a. Has a keyword *abstract* before the keyword class.
  b. Has a method marked as *abstract*. This method is called an abstract method.
  c. The abstract method always ends with a semicolon (;). It has no content.
- To explain your TA, students SHOULD try to understand the following **example**:

```
public abstract class GeometricObject {
	public void printName() {
		System.out.println("GeometricObject");
	}

	//abstract method
	public abstract double findArea();

	public abstract double findPerimeter();
}
```

## 4.5 Rules on Abstract classes

- Abstract classes are like regular classes with data and methods, but students cannot create instances of the abstract class using the new operator
  - To explain your TA, students SHOULD try to understand the following **example**:
    `new Geometry ();`

- Students can use the abstract class as a type.
  - To explain your TA, students SHOULD try to understand the following **example**:
    `Geometry geometry01;`
    A class that contains an abstract method must be an abstract class.
    However, an abstract class doesn't necessarily contain the abstract methods

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

## 4.6 Interfaces

- An interface is a class like structure that contains only constants and abstract methods.
- Java uses the keyword *interface* to distinguish it from a class.
- Also since the interface is known by its definition to contain only the constants and the abstract methods, the keyword abstract in front of the method may be omitted.
- Variables in an interface must be public, static, and final.
- To explain your TA, students SHOULD check the following properties to understand the interface class in Java code:
    1. No constructor.
    2. Cannot be instantiated using the new.
    3. Not contain any constructors.
    4. Not extended by a class; it is implemented by a class.

## 4.7 When a Class Derives from an Interface?

- When a class derives a parent which is an interface, students MUST use the keyword *`implements`* instead of *`extends`*.
- To explain your TA, students SHOULD check the following properties to understand deriving a Java class from the interface class in Java code:
    1. A class can derive from multiple interface parents.
    2. The comma (,) is used for listing interface parents
    3. A non-interface parent must be listed before interface parents

    4. An interface can inherit other interfaces using the *extends* keywords. Such an interface is called a subinterface.
- To explain your TA, students SHOULD try to understand the following **example**:

```
public interface Edible {
	public String howtoEat();
}

public abstract class Fruit implements Edible {
	public double CalculateVitC();
	public String howToEat() {
		return “Make Fruit Juice”;
}
```

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

# [DES103(Y23S2)-LAB04-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 Classroom.

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

<StudentID>_<Lab number>_<Exercise name>

If your student's ID is 6722300208, the name format of your java project should be:

6722300208_LAB04_TaxPayment

## Exercise 1: (1 point)

Project Name : <StudentID>_LAB04_TaxPayment
Instruction : Download and complete an abstract class `Employee` from the following UML.
Remark : *+ denotes public modifier.*

| ***Employee*** | **Description** |
|---|---|
| `name: String`<br>`position: String` | Name of Employee<br>Position of Employee |
| `Employee(name: String, position: String)`<br><br>*`calculateMonthlyIncome( )`*`: double`<br><br>*`printWorkPlace( )`*`: void`<br><br>`printInfo(): void`<br><br>`calculateTaxRate(salary: double): double` | A 2-arguments constructor<br><br>An abstract method to calculate and return the monthly income of an employee<br><br>An abstract method to print `WorkPlace` for its subclass<br><br>Print info of an `Employee` in the following format<br>`name is a position`<br><br>Calculate and return the tax rate of `Employee` from the input salary in the following conditions.<br>• Less than or equal to 200000, the tax rate is 0.0<br>• Less than or equal to 400000, the tax rate is 0.05<br>• Less than or equal to 600000, the tax rate is 0.1<br>• Less than or equal to 800000, the tax rate is 0.15 |

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

| | |
|---|---|
| | • Otherwise, the tax rate is 0.2<br><br>An abstract method to calculate and return the monthly income of an employee<br><br>An abstract method to print `WorkPlace` for its subclass |

## Exercise 2: (1 point)

Project Name : <StudentID>_LAB04_ TaxPayment

Instruction : Download and complete an interface `TaxPayer` from the following UML.

Remark : *+ denotes public modifier.*

| **<<interface>> TaxPayer** | **Description** |
|---|---|
| | |
| `+ calculateYearlyIncome( ): double`<br>`+ calculateTax( ): double`<br>`+ payTax( ): void` | An abstract method for calculating yearly income<br><br>An abstract method for calculating Tax<br><br>An abstract method for paying Tax |

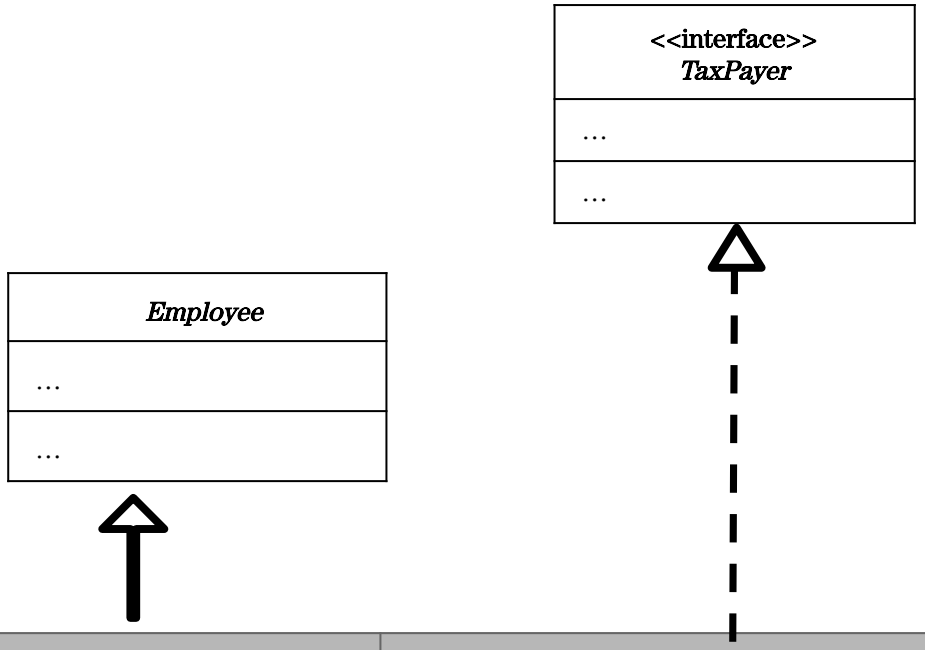
Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

## Exercise 3: (3 points)

Project Name : <StudentID>_LAB04_ TaxPayment

Instruction : Download and complete a class `FullTime` as a subclass of `Employee` and `TaxPayer`.

Remark : *+ denotes public modifier.*




| **FullTime** | **Description** |
|---|---|
| `workplace: String`<br>`salary: double` | workplace of Employee<br>salary of Employee |
| `FullTime(name: String,`<br>`position: String,`<br>`workplace: String,`<br>`salary: double)` | A 4- arguments constructor |
| `void printWorkPlace( )` | Overridden method to print workplace of an Employee in the following format:<br>`Work at workPlace` |
| `calculateMonthlyIncome( ): double` | Overridden method to calculate and return salary |
| `+calculateYearlyIncome( ): double` | Overridden method to calculate and return 12*salary |
| `+calculateTax( ): double`<br><br>`+ payTax( ): void` | Overridden method to calculate and return tax value which is the tax rate * YearlyIncome<br>**Remark**: use the `calculateTaxRate()` method of the employee to obtain the tax rate |

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

| | |
|---|---|
| `+ printInfo( ): void` | Overridden method to print the output in the format of<br>`Pay tax $CalculatedTax`<br><br>Overwritten method to print the `FullTime` in the following format<br>`name is a position`<br>`Work at workPlace`<br>`Yearly income is calculatedYearlyIncome`<br>`Pay tax $CalculatedTax` |

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

## Exercise 4: *(3 points)*

Project Name : <StudentID>_LAB04_ TaxPayment

Instruction : Download and complete a class `PartTime` as a subclass of `Employee` and `TaxPayer`.

Remark : *+ denotes public modifier.*

| PartTime | Description |
|---|---|
| `workplace: String`<br>`numHrPerWeek: double`<br>`hourlyRate: double` | workplace of Employee<br>a number of working hours per week an hourly rate |
| `PartTime(name: String,`<br>`position: String,`<br>`workplace: String,`<br>`numHrPerWeek: double,`<br>`hourlyRate: double)` | A 4-arguments constructor that initialize the properties of the superclass and properties of PartTime |
| `printWorkPlace( ): void` | Overridden method to print workplace of a PartTime in the following format:<br>`Work at` |
| `calculateMonthlyIncome( ): double` | Overridden method to calculate and return numHrPerWeek*hourlyRate*4 |
| `+calculateYearlyIncome( ): double` | Overridden method to calculate and return YearlyIncome which is 12 * monthly income |
| `+calculateTax( ): double` | Overridden method to calculate and return tax value which is the tax rate * YearlyIncome<br>Remark: use the method calculateTaxRate( ) of the employee to obtain the tax rate |
| `+payTax( ): void` | Overridden method to print the output in the format of<br>`Pay tax` |
| `printInfo( ): void` | Overwritten method to print the FullTime in the following format:<br>`Name is a position`<br>`Work at workPlace`<br>`Yearly income is XXXXX`<br>`Pay tax $XXXXX` |

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

## Exercise 5: *(2 points)*

Project Name : <StudentID>_LAB04_TaxPayment

Instruction : Download and complete a class `TaxPaymentTest` with a `main` method.
In main, write the code that complete the following task.

Remark : *+ denotes public modifier.*

a) Download the class `TaxPaymentTest` with the `main` method and run the program. If you write the programs `TaxPayer`, `Employee`, `FullTime` and `PartTime` correctly, the outputs should be as follows.

```
Erika Parker is a secretary
Work at InfoTech
Yearly income is 390000.0
Pay tax $19500.0
--------------------

Brian Lee is a driver
Work at InfoTech
Yearly income is 72000.0 Pay tax $0.0
--------------------

James Knein is a technician
Work at InfoTech
Yearly income is 426000.0
Pay tax $42600.0
--------------------

Anne Lin is an accountant
Work at InfoTech
Yearly income is 216000.0
Pay tax $10800.0
--------------------

Jamie Fox is a manager
Work at InfoTech
Yearly income is 687600.0
Pay tax $103140.0
--------------------
```

b) Uncomment the codes in **this part b),** there are errors in these statements.
Can you explain the reason?
Answer this question to the TA to get a pass on this problem.
Without changing the class-types declaration of the following variables:
`employee03`, `employee04`, and `employee05_TaxPayer`,
what do you need to do to be able to call `payTax()` of `employee03`, `employee04`, and `employee05_TaxPayer`. (Hint: casting)
The outputs should be as follows.

```
Pay tax $42600.0
Pay tax $10800.0
Pay tax $103140.0
```