Module: Java + UI + PHP

Course: Core Java

Session 10: Abstract Classes and Interfaces

Trainer Notes

1 Session Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Time  (min) | Content | Methodology | Trainer  Approach | Learner  Activity | Learning  Outcome  (Bloom's) | Learning  Outcome  (Gardner's) |
| 15 | Abstract classes | Reference to  Reading  Material and  Slides | Facilitate,  Elicit  responses | Think,  Respond,  Identify | Remember,  Understand | Intrapersonal,  interpersonal |
| 15 | Abstract classes demo | Reference to  Reading  Material and  Slides | Facilitate,  Elicit  responses | Think,  Respond,  Identify | Remember,  Understand | Intrapersonal,  interpersonal |
| 15 | Interfaces | Reference to  Reading  Material and  Slides | Facilitate,  Elicit  responses | Think,  Respond,  Identify | Remember,  Understand | Intrapersonal,  interpersonal |
| 15 | Interfaces demo | Reference to  Reading  Material and  Slides | Facilitate,  Elicit  responses | Think,  Respond,  Identify | Remember,  Understand | Intrapersonal,  interpersonal |
| 25 | Guided Classroom  Activities | Group Activities | Facilitate | Work on guided activities | Remember,  Understand,  Coding | Intrapersonal,  interpersonal |
| 05 | Conclusion | Discussion | Question,  Facilitate,  Guides | Participates,  Recollect  concepts | Remember | Intrapersonal,  interpersonal |

2 Objectives

* Explain abstract class
* Explain why abstract classes are needed
* Demonstrate use of abstract classes in writing small java applications
* Explain Interface
* Demonstrate interfaces in writing small java applications
* Discuss and demonstrate “final” keyword

3 Materials Needed

* Slides

1. Presentation Description

The Facilitator is expected to follow the Presentation Slides as a guideline for the flow of the session.

1. **Classroom Activities**

Abstract Class

To make the Employee class un-instantiable.

The payroll system of an organization involves calculating the gross salary of each type of employee and the tax applicable to each. The entity classes, their fields and methods are already given in your candidate project. Your task today is to change the Employee class to have no instances.

Classes given to you in the candidate project are: Employee, Trainer, Sourcing, Manager, Organization and TaxUtil. These classes already have fields, constructors and the respective methods. The Trainer, Manager and sourcing classes are sub-classes of Employee.

In many real world applications, we often find entities that are common names for specific instances. For example we use the common name “car” for many brands of cars available. But there is no auto mobile that is just a “car”. It IS a car, but, from a particular manufacturer and of a particular model. Similarly in our application, Manager, Trainer, Sourcing are all designations of employees, but there is no employee without a designation. In such instances, it makes no sense to create objects of the Employee class, as there is no one in the organization who is just an employee, without designation.

Change the Employee class such that you cannot instantiate it, BUT it should remain as the base class for the other classes already derived from it.

Note that the addEmployeeBulk method in Organization that was provided to you for initial population has already been changed to use Manager instead of Employee class.

Interfaces

To create an interface and use it in the classes of the Employee payroll system

The payroll system of an organization involves calculating the gross salary of each type of employee and the tax applicable to each. The entity classes, their fields and methods are already given in your candidate project. Your task today is to create an interface for income calculation.

Classes given to you in the candidate project are: Employee, Trainer, Sourcing, Manager, Organization,SelfEmployed and TaxUtil. These classes already have fields, constructors and the respective methods. The Trainer, Manager and sourcing classes are sub-classes of Employee. The method calculateGrossSalary has been removed from all the entity classes as this method is no longer necessary.

Some methods are generic and may be used by many unrelated classes. Such methods are defined in an interface and classes implement these interfaces.

* Define an interface called IncomeCalculatorInterface in the entity package. In this Interface, define the method:public double calculateIncome().
* The classes Employee must implement this interface. You will now have to define the methodcalculateIncome() in the sub classes of Employee class.
* The calculation of income for the different classes are as given below. Use this logic in the calculateIncome() method:
  1. Class Manager-Calculate the Income as : basicSalary +HRAPer +DAPer + projectAllowance
  2. Class Trainer-Calculate the Income as : basicSalary +HRAPer +DAPer +(batchCount \* perkPerBatch)
  3. Class Sourcing-Calculate the Income as : basicSalary +HRAPer +DAPer +((enrollmentReached/enrollmentTarget)\*100)\*perkPerEnrollment)
* Since the calculateGrossSalary method is now removed from Employee and its sub classes, the method calculateTax in TaxUtil needs to be changed. It should now accept the Interface you defined as a parameter instead of the Employee class. Change tis signature as follows:

1.public double calcuateTax(IncomeCalculatorInterface incomeCalc) - This method should calculate the tax using the logic as given below: 2. if totalIncome < 30000, tax is 5% of taotal Income, otherwise it is 20% of total Income. 3. This method should now call the calculateIncome() method of the IncomeCalculatorInterface instead of the calculateGrossSalary method to get the gross Income.

* A new class called SelfEmployed is given to you in the entity package. This class represents those people who are self-employed. It contains two fields totalIncome and totalExpense. Change this class to implement the Interface IncomeCalculatorInterface that you defined in step1.
* You will now have to define the method calculateIncome for this class as well. The income can be calculated as: totalIncome-totalExpense.
* Verify that the calculateTax method of TaxUtil class will now also work for SelfEmployed class as well.