**OOP Lab 5**

|  |  |  |  |
| --- | --- | --- | --- |
| Name: |  | Department: |  |
| Student ID: |  | Room Number: |  |
| Due Date: | **April 6, 23 : 59** | | |

* **Submit your assignment using the following file format:**

LabNumber\_StudentName.zip (eg. Lab5\_Hongkildong.zip).

* This zip file will contain **two types of** files, namely:

1. **report file** with file format **“Report\_Lab number**” (eg. report\_5) to answer theory questions and to write the screen shot of your program.
2. Source code file that contains codes of classes to answer programming questions.
3. **Objectives**

1. Learning the concept of an **implicit parameter** called **“this”** reference keyword

2. Learning the concept of **Overloaded constructor** using “**this**”refrence keyword.

3. Learning and utilizing the concept of **composition (HAS-A relationship between two** classes).

4. Learning and utilizing the use of “**final**” member variable,

5. Learning the similarity and difference between inheritance and polymorphism, and how to utilize them.

6. Learning the similarity and difference between method overriding (dynamic binding) and method **overloading (static binding) in terms of polymorphism.**

**7. Learning similarity and difference between of concrete class and abstract class and how to utilize them**

**8. Learning the similarity and difference between abstract class and interface and how to use them**.

**II. Exercises (31 points)**

1. Answer the following questions about “**ThisTest**” class and “**SimpleTime**” class (**Fig.8.4).**
   1. “**this.toUniversalString**()” and “**toUniversalString**()” within **buildString(**) method have the same results. Why they have the same result? Explain only the reason (**2 point**)
   2. Remove all “**this”** reference variables in **SimpleTime** class. Re-write the code to get the same result as shown in (A) (include your code and captured result screen)(**2 point**)
2. Answer the questions about **Account** class of **Fig 3.8** and **AccountTest class** of **Fig 3.9.**
   1. Add the following **constructor** in the **Account class**. This constructor should call an **original** **constructor** of the Account class (include the code of the added constructor code)(**2pt**).

|  |
| --- |
| Account (String name); **//**  initializing **name** by the given String and balance by zero.  {  ………….  } |

* 1. Using the code in (A), edit 10th line of **AccountTest class** as shown below, and then execute the program (including capture result screen) (**2 point**)

|  |
| --- |
| Account account2 = **new** Account("Hong Kil-Dong"); |

1. Answer the questions using **Date**, **Employee**, **EmployeeTest** classes in **Fig** **8.7 /8.8 / 8.9**.
   1. Once “**birthDate**” and “**hireDate**” member fields of the Employeeclass of Fig 8.8 are initialized, you don’t need to re-initialize them again. Execute the program after adding “**final modifier**” on two member variables. **Check what results happens before and after**. Explain the reason (including result difference before and after, and reason) (**2 point**)
   2. Once “birthDate” and “hireDate” member fields of the Employee class of Fig 8.8 are initialized, you don’t need to re-initialize them again. **In the Employee class**,

* add “final modifier” on two member variables.
* Add “SetBirthDate() and SetHireDate() methods and try updates the initial values of “birtgDate” and “hireDate” fields by adding new values.

1. What happens when you try to implement the above two methods?

(including your reason) (2 point)

* 1. After running the previous program in (A), add the following constructor in the ***Employee*** class. What result you get? Explain your reason). (2 point)

|  |
| --- |
| **Public Employee**() { } |

1. **Answer the following questions using CommissionEmployee (Fig 9.10), BasePlusCommissionEmployee (Fig 9.11) and PolymorphismTest (Fig 10.1).**
   1. Find all places where polymorphism is used in Fig 10.1. (2 points).
   2. After line 34 in Fig 10.1, insert the following code and run the program. What happens? Why does that happen? (2 points)

|  |
| --- |
| BasePlusCommissionEmploye **bpce** = commissionEmployee2;  System.out.println("Salary = " + **bpce.getBaseSalary**()); |

* 1. Replace the code in B by the following code and run the program. What is different between program in A and program B ? (2 points)

|  |
| --- |
| BasePlusCommissionEmployee **bpce** = (BasePlusCommissionEmployee) commissionEmployee2;  System.out.println("Salary = " + **bpce.getBaseSalary**()); |

* 1. Replace the code in C by the following code and run the program. What happens? Why does it happen? (2 points)

|  |
| --- |
| BasePlusCommissionEmployee **bpce** = (BasePlusCommissionEmployee) commissionEmployee;  System.out.println("Salary = " + **bpce.getBaseSalary**()); |

1. **Answer the questions after running Payroll system in Section 10.5 (Fig 10.4 ~ 10.9, Employee, SalariedEmployee, HourlyEmployee, CommissionEmployee, BasePlusCommissionEmploye, PayrollSystemTest).**
   1. The **output** of the program has **two** different parts. Explain the **output** of each part (**2 points**)
   2. Change the code in **Employee** class as follows,

* deleting “**abstract**” keyword at line **4:** **public class Employee**
* Change line **46** by the code**: public double earnings() { return 0.0; }**

After running the program, compare the results with previous (unchanged) program (**1 points**)

* 1. **Before changing** and **after** changing in (B), add the following code to the **main** function and run it.

|  |
| --- |
| Employee e = **new** Employee("Kildong", "Lee", "000-00-0000"); |

What is the difference between the two? Explain the reason (2 points)

1. **Answer the questions after running Payroll system in Section 10.9 (Fig 10.11 ~ 10.15, Payable, Invoice, Employee, SalariedEmployee, PayableInterfaceTest).**
   1. Modify the program by adding the following **subclasses** of the Employee class (see Fig.10.2 and Fig.10.10) (**2pt)**.

* **CommissionEmployee**
* **HourlyEmployee**
* **BasePlusCommissionEmployee**( Subclass of Commission Employee)
  1. **Modify the code** of the new classes in A **by** replacing their **earnings (**) methods using **getPaymentAmount()** method similar to the **SalariedEmployee class**. Modify also **PayableInterfaceTest** class by **expanding the array**, so that the class can create one object of each subclass and calculate the payment (**2 points**)

1. **Dog, Cat , and Sheep are all animals with their own unique cries. Answer the following questions.**
   1. Define a class of each animal including at least **one method** as follows.

|  |
| --- |
| void **cry**()  {  ...  } |

However, the crying operation is replaced by printing a string corresponding to the crying sound (2point).

**Example:**  sound of dog: “waw waw waw”

**Example:** sound of Cat: “ Miyaw Miyaw Miyaw”

**Example:** sound of sheep: “Ba Ba Ba ”

* 1. Run **AnimalTest** class that tests the crying of these animals using both non-polymorphism and **polymorphism approaches** (2 points)