ALGONQUIN COLLEGE

OBJECT ORIENTED PROGRAMMING (JAVA) (CST8284)

LAB 6

Polymorphism

Overview of Lab 6

- This lab focuses on concepts taught in this course especially: **abstract classes, inheritance and polymorphism.** You will deploy your knowledge of these concepts to implement polymorphic behavior in given scenarios.

- A superclass named **Programmer** has been extended by several subclasses (see the code files provided) and a given method (**earnings**) is required to be implemented polymorphically across classes.

Before You Start…

- Several code files have been provided to you. Review **all** these files to understand how the classes connect, the logic and the hierarchy among the given classes.

- Do NOT proceed with providing solutions to the tasks until you have reviewed each code file.

- Examine your tasks as stated **in this slide deck** to ensure you are doing the right thing.

PART 1

ALGONQUIN COLLEGE

Part 1 Items provided for you:

- Review all **7 Java code files and sample output** provided for you in this lab.
  - **Programmer.java** (programmer working in a company)
  - **Date.java** (declaring the date – useful for birthday of a programmer)
  - **MyPaySystemTest.java** (provides the main method to test your classes)
  - **CommissionProgrammer.java** (programmers who are paid based on commission)
  - **HourlyProgrammer.java** (programmers who are paid per hour)
  - **SalariedProgrammer.java** (regular salaried programmers)
  - **BasePlusCommissionProgrammer** (extends CommissionProgrammer)

Your Tasks…

1. Draw the **UML diagram** for the classes and subclasses based on the hierarchy you have observed in the code files given to you.

2. In the **Programmer** class provided, make a change to this class by **inserting** a private instance variable named **birthDate**.

3. Scroll down the **Programmer** class to see the **"TO DO"** area and supply the missing code.

Your Tasks… (2)

4. Use the **Date** class provided to you to represent the **birthday** of a programmer.

5. Inspect the **Date** class provided. Add ***get* methods** to this class (check to see **"TO DO: Complete this portion"** area in the code for your action).

Note that the programmers irrespective of what form of compensation they have are **paid** only **once every month.**

Your Tasks… (3)

6. Create an **array** of Programmer variables in MyPaySystemTest class, which should store the references to the various **programmer objects**.

7. **Use a loop** to compute the payroll for each programmer (**polymorphically**).

8. Implement the condition that for the **current month** in which the **birthday** of any programmer occurs. **Add** a **bonus of $500.00** to the payroll amount of that programmer.

Your Task… (4)

9. Update the **MyPaySystemTest** class provided for you ensuring that the user can be prompted to **enter the current** month.

10. **Check all the files to see portions to complete** that has been marked for you to **insert your code in all classes**. Complete all for your code to run.

11. Check the output sample to ensure that your output follows the same pattern **(part 1 is provided). Part 2 (which you should provide as well, will differ)**

Demo PART 1 – What will be graded

- Show your Programmer the hierarchy UML diagram
- Run your files to show that they work
- Show your output file to your Professor
- Show your professor the updates you made to **Programmer.java, Date.java** and **MyPaySystemTest.java**
- Prepare to answer some questions.

PART 2

ALGONQUIN COLLEGE

PART 2: Your Tasks…

1. Provide an updated **UML class diagram** that includes the new class based on the hierarchy you have observed in the code files given to you.
2. **Update the existing payment system:**

- Create a new subclass named **ActionProgrammer** that extends **Programmer**. This new subclass represents a programmer whose pay depends on the number of **apps** he creates while on duty.
-

PART 2: Your Task…(2)

3. The new subclass **ActionProgrammer,** contains:

- private instance variable wage which stores the **wage** of the programmer per piece (each) of app created.
- private variable pieces, which stores the number of **pieces** of apps created.

PART 2: Your Task…(3)

4. Provide a concrete implementation of method **earnings** in the new class **ActionProgrammer** that will compute the earnings of the programmer given by:

(Num. of pieces made * wage per piece)

That is: the number of pieces of apps created multiplied by the wage for each piece.

PART 2: Your Task…(4)

- Now, create an array of **Programmer variables** to store the references to objects of each concrete class in your updated **Programmer** hierarchy.

- This should show the String representation and **each of the existing Programmer** earnings (hint: combine all from part 1 and part 2).

- Create a **new test class** to test the updated system (new hierarchy), and save it as **MyPaySystemTest2.java**

Demo PART 2: What will be graded

- Show the updated hierarchy UML class diagram (which now includes ActionProgrammer)
- Show the **ActionProgrammer** class you created
- Show the new launcher/driver class you created named **PayrollSystemTest2**.