



LAB 05 Polymorphism-Interface

- Submit **whole java project file in zipped format** and named project file as *NameSurname_studentNo_Lab5*
- Do your homework in **ECLIPSE IDE**.
- Do not use Turkish Characters(ç,ğ,ı,ö,ş,ü) for naming project, methods, classes.
- Late submissions are not allowed.
- You should do homework **YOURSELF**. Group working is not allowed.
- Copy homework will be evaluated as 0.
- Use Google Classroom for your questions.

NOTE: Use true encapsulation (design all data fields as private and reach them just using get/set methods) and inheritance practice in your implementation. Your project have to consist of 8 classes: *Payable*, *Invoice*, *Employee*, *SalariedEmployee*, *HourlyEmployee*, *CommisionEmployee*, *BasePlusCommisionEmployee*, *PayableInterfaceTest*

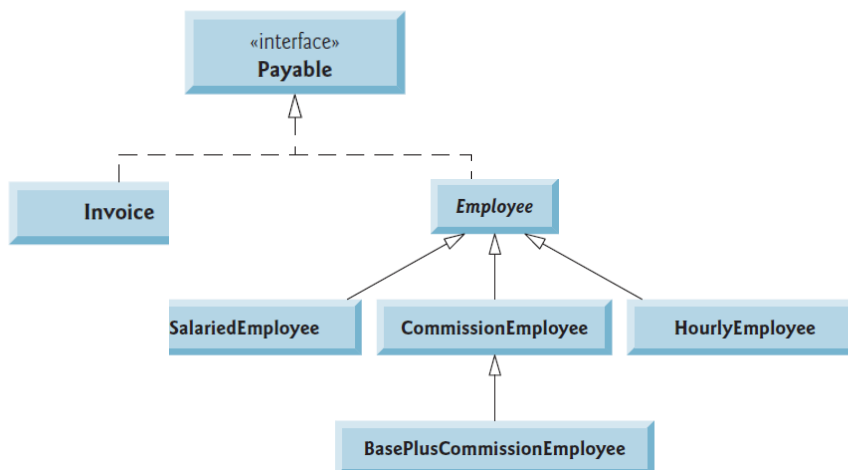
NOTE: Do not use *earnings()* method for this homework instead, *getPaymentAmount()* of *Payable* interface will be implemented.

NOTE: **Do not take input from user!!**

HOMEWORK

- ✚ CEO of Factory X make a request to your software company about design a program that calculates Factory X's payment amount which will be made for employees and invoices. Create **Invoice**, **Payable** (interface), **Employee**, **SalariedEmployee**, **HourlyEmployee**, **CommisionEmployee**, **BasePlusCommisionEmployee**, and **Test** (main) class according to requirements given below.

Hierarchy between classes:





ADNAN MENDERES UNIVERSITY

CSE 203 Object-Oriented Programming

- **Invoice** implements **Payable**
`public class Invoice implements Payable`
- **Employee** implements **Payable**
- **Employee** is abstract superclass. Each employee has a **first name**, a **last name** and a **social security number**. There is no `getPaymentAmount()` method in **Employee** class
`public abstract class Employee implements Payable`
- **Salaried Employee**, **Hourly Employee**, **Commission Employee** is an **Employee**.
- **BasePlusCommissionEmployee** is a **Commission Employee**.
- Each class has different behavior in `getPaymentAmount()` and `toString()` methods. **NOTE:** There is no `getPaymentAmount()` method in **Employee** class.

Factory X's payment amount which will be made for its employees on a weekly basis. The employees are of four types:

- **Salaried employees** are paid a fixed weekly salary regardless of the number of hours worked
- **Hourly employees** are paid by the hour and receive overtime pay (i.e., 1.5 times their hourly salary rate) for all hours worked in excess of 40 hours
- **Commission employees** are paid a percentage of their sales
- **Base plus commission employees** receive a base salary plus a percentage of their sales.

Factory X's payment amount which will be made for invoice:

- **Invoice** 's payment amount is calculated by multiplying quantity and price per item.

	<code>getPaymentAmount()</code>	<code>toString()</code>
Employee	Not implemented	<i>firstName lastName</i> social security number: <i>SSN</i>
Salaried- Employee	<i>weeklySalary</i>	salaried employee: <i>firstName lastName</i> social security number: <i>SSN</i> weekly salary: <i>weeklySalary</i>
Hourly- Employee	<pre> if (hours <= 40) wage * hours else if (hours > 40) { 40 * wage + (hours - 40) * wage * 1.5 } </pre>	hourly employee: <i>firstName lastName</i> social security number: <i>SSN</i> hourly wage: <i>wage</i> ; hours worked: <i>hours</i>
Commission- Employee	<i>commissionRate * grossSales</i>	commission employee: <i>firstName lastName</i> social security number: <i>SSN</i> gross sales: <i>grossSales</i> ; commission rate: <i>commissionRate</i>
BasePlus- Commission- Employee	<i>(commissionRate * grossSales) + baseSalary</i>	base salaried commission employee: <i>firstName lastName</i> social security number: <i>SSN</i> gross sales: <i>grossSales</i> ; commission rate: <i>commissionRate</i> ; base salary: <i>baseSalary</i>



ADNAN MENDERES UNIVERSITY

CSE 203 Object-Oriented Programming

CLASSES

Payable

- Design an interface named **Payable** with a method: **double getPaymentAmount()**.

```
public interface Payable
{
    double getPaymentAmount();
}
```

Invoice

```
public class Invoice implements Payable
```

- Create a class named **Invoice** contains billing information for only one kind of part:
- implements **Payable**
- private **partNumber**, **partDescription**, **quantity** and **pricePerItem** data fields
- 4 argument Constructor:**
Validation: **quantity** must be ≥ 0 , price must be > 0
Make necessary assignments.
- get/set methods for partNumber, partDescription, quantity and pricePerItem**
Apply validation in set methods.
- toString()** method returns string representation of an object.
- Implement **Payable's getPaymentAmount()** that returns multiplication of **quantity** and **pricePerItem**.

Employee

- Employee** is abstract superclass.
- implements **Payable**
- private **firstName**, **lastName** and **social security number (SSN)** data fields.
- 3 argument Constructor**
- get/set methods for firstName, lastName and social security number (SSN)**
- toString()** method returns string representation of an object.
- getPaymentAmount()** method of **Payable interface** should not be coded in Employee class.
The method content changes employee to employee so, **it should be coded in all subclasses of Employee and Invoice**
- Employee** class must be declared **Abstract** to avoid compilation error because **getPaymentAmount()** is not implemented in the class.

SalariedEmployee

- Subclass of **Employee**
- private **weeklySalary** data field
- 4 argument Constructor:** Use **super** keyword when invoking superclass constructor,
Validation: **weeklySalary** must be ≥ 0
Make necessary assignments.
- get/set methods for weeklySalary**
Apply validation in set method.



ADNAN MENDERES UNIVERSITY

CSE 203 Object-Oriented Programming

- Implement **getPaymentAmount()** and override **toString()** methods according to table above. Use **super** if necessary

HourlyEmployee

- Subclass of **Employee**
- private **wage, hours** data fields
- **5 argument Constructor:** Use super keyword when invoking superclass constructor, **Validation:** wage must be ≥ 0 , hours must be ≥ 0 and < 168
Make necessary assignments.
- **get/set methods for wage, hours**
Apply validation in set methods.
- Implement **getPaymentAmount()** and override **toString()** methods according to table above. Use **super** if necessary

CommissionEmployee

- Subclass of **Employee**
- private **grossSales, commissionRate** data fields
- **5 argument Constructor:** Use super keyword when invoking superclass constructor, **Validation:** grossSales must be ≥ 0 , commissionRate > 0 and < 1
Make necessary assignments.
- **get/set methods for grossSales, commissionRate**
Apply validation in set methods.
- Implement **getPaymentAmount()** and override **toString()** methods according to table above. Use **super** if necessary

BasePlusCommissionEmployee

- Subclass of **CommissionEmployee**
- private **baseSalary** data field
- **6 argument Constructor:** Use super keyword when invoking superclass constructor, **Validation:** baseSalary must be ≥ 0
Make necessary assignments.
- **get/set methods for baseSalary**
Apply validation in set methods.
- Implement **getPaymentAmount()** and override **toString()** methods according to table above. Use **super** if necessary

Test (Main class)

- You may use your own preferred data field values while creating objects.
- Create Payable type array which contains 6 elements.
`Payable payableObjects[] = new Payable[6];`
- Initialize array with references of 2 **Invoice** object, a reference of **SalariedEmployee**, **HourlyEmployee**, **CommissionEmployee**, **BasePlusCommissionEmployee** respectively.

```
payableObjects[ 0 ] = new Invoice(...);  
payableObjects[ 1 ] = new Invoice(...);  
payableObjects[ 2 ] = new SalariedEmployee(...);  
payableObjects[ 3 ] = new HourlyEmployee(...);  
payableObjects[ 4 ] = new CommissionEmployee(...);  
payableObjects[ 5 ] = new BasePlusCommissionEmployee(...);
```



ADNAN MENDERES UNIVERSITY

CSE 203 Object-Oriented Programming

Polymorphic Usage of Interface:

- For each element in the array;
 - Print the element
 - Decide if element is a BasePlusCommissionEmployee or not
`if (payableObjects[i] instanceof BasePlusCommissionEmployee)`
 - If so, increase baseSalary of employee by adding %10 of current baseSalary and print new baseSalary
 - Print getPaymentAmount()
- Print the class name of each object in the array using default superclass Object's related method.
`payableObjects[j].getClass().getName()`

Sample run :

----- Invoices and Employees processed polymorphically:-----

invoice:

part number: 01234 (seat)
quantity: 2
price per item: \$375,00
payment amount: \$750,00

invoice:

part number: 56789 (tire)
quantity: 4
price per item: \$79,95
payment amount: \$319,80

salaried employee: John Smith

social security number: 111-11-1111
weekly salary: \$800,00
payment amount: \$800,00

hourly employee: Karen Price

social security number: 222-22-2222
hourly wage: \$16,75; hours worked: 40,00
payment amount: \$670,00

commission employee: Sue Jones

social security number: 333-33-3333
gross sales: \$10.000,00; commission rate: 0,06
payment amount: \$600,00

base-salaried commission employee: Bob Lewis

social security number: 444-44-4444
gross sales: \$5.000,00; commission rate: 0,04; base salary: \$300,00
new base salary with 10% increase is: \$330,00
payment amount: \$530,00

Payable object 0 is a Invoice

Payable object 1 is a Invoice

Payable object 2 is a SalariedEmployee

Payable object 3 is a HourlyEmployee

Payable object 4 is a CommissionEmployee

Payable object 5 is a BasePlusCommissionEmployee