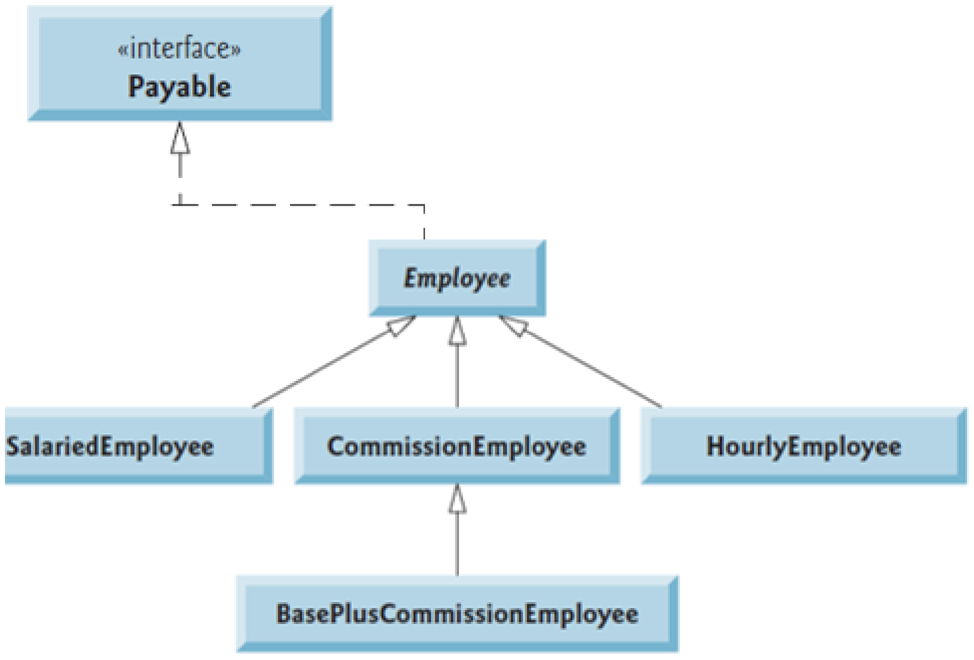
ADNAN MENDERES UNIVERSITY
CSE 203 Object-Oriented Programming

Project

- You should submit **whole java project file in zipped format** and **project report,** named project file as ***NameSurname_studentNo_Project***. Project report should be in .doc, .docx or .pdf format. Name of one student is enough for naming project
- Do your homeworks in **ECLIPSE IDE**.
- Do not use Turkish Characters(ç,ğ,ı,ö,ş,ü) for naming project, methods, classes.
- Late submissions are not allowed.
- Group working is allowed **up to 4 members**. **Indicate all project members's student No-Name surname in the report.**
- Copy homework will be evaluated as 0.
- You have to use JavaFX for your user interface design. Do not use "Scene Builder". Design your user interface by coding.
- Use Google Classroom for your questions.

Problem Description:
CEO of Factory X make a request to your software company about design a program with user interface. The program should calculate Factory X's payment amount which will be made for all type of employees. Create **Payable** (interface), **Employee**, **SalariedEmployee**, **HourlyEmployee**, **CommissionEmployee**, **BasePlusCommissionEmployee,** and **Test (main)** class.

Hierarchy between classes:

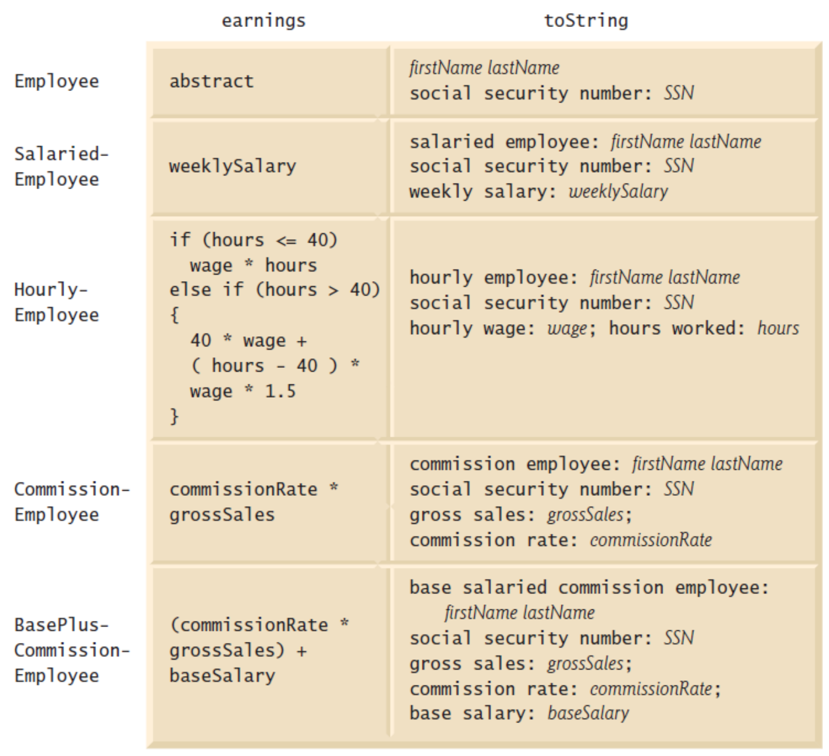


- ✓ **Employee** implements **Payable**
- ✓ **Employee** is abstract superclass. Each employee has a **first name**, a **last name** and a **social security number**. There is no `getPaymentAmount()` method in Employee class
  ```
  public abstract class Employee implements Payable
  ```
- ✓ **Salaried 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:



	earnings	toString
Employee	abstract	*firstName lastName* social security number: *SSN*
Salaried-Employee	weeklySalary	salaried employee: *firstName lastName* social security number: *SSN* weekly salary: *weeklySalary*

Hourly-Employee	if (hours <= 40)
	wage * hours
	else if (hours > 40)
	{
	40 * wage +
	(hours - 40) *
	wage * 1.5
	}
		hourly employee: *firstName lastName*
		social security number: *SSN*
		hourly wage: *wage*; hours worked: *hours*

Commission-Employee	commissionRate * grossSales	commission employee: *firstName lastName* social security number: *SSN* gross sales: *grossSales*; commission rate: *commissionRate*
BasePlus-Commission-Employee	(commissionRate * grossSales) + baseSalary	base salaried commission employee: *firstName lastName* social security number: *SSN* gross sales: *grossSales*; commission rate: *commissionRate*; base salary: *baseSalary*

CLASSES

Payable

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

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

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.
- 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

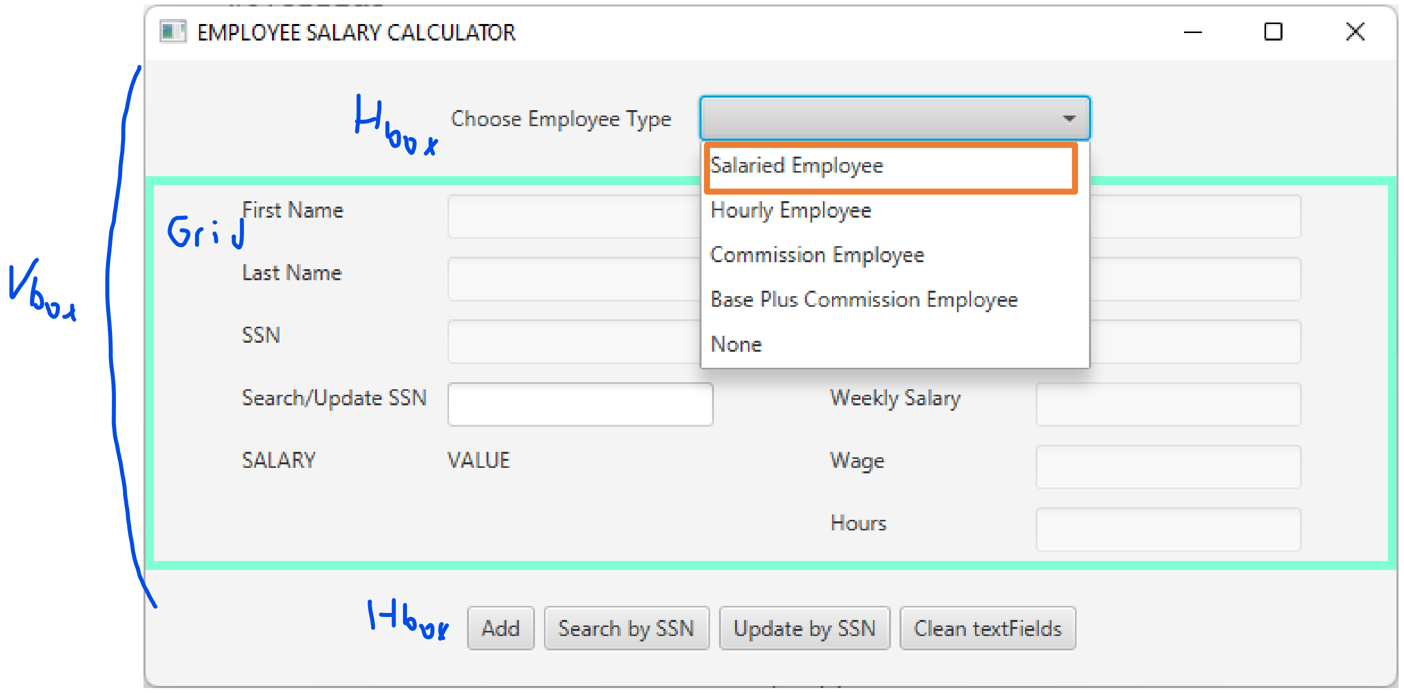
- Subclass of **Employee**
- private **grossSales**, **commissionRate** data fields
- **5 argument Constructor:** Use super keyword when invoking superclass constructor, **Validation:** grossSales must be >=0, commissionRate>0 and <1
  Make necessary assignments.
- **get/set methods for grossSales, commissionRate**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

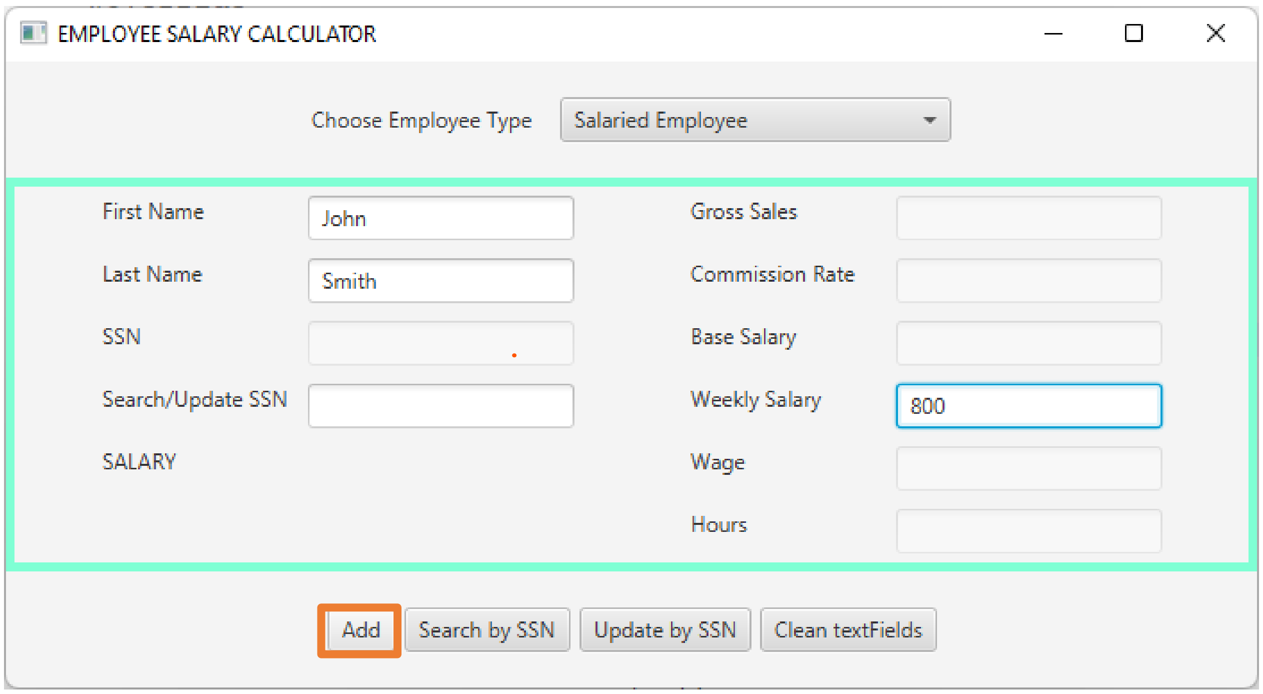
BasePlusCommissionEmployee

- 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

PROJECT SCREENSHOTS AND REQUIREMENTS





EMPLOYEE SALARY CALCULATOR

Choose Employee Type: Salaried Employee

First Name	John	Gross Sales	
Last Name	Smith	Commission Rate	
SSN		Base Salary	
Search/Update SSN		Weekly Salary	800
SALARY		Wage	
		Hours	

1- Your program should store, add, update and search salary information of all type employees as shown in figures above. Use **random access file(.dat) or text file (.txt)** for reading and writing an employee information.

- ✓ Program should perform **Add**, **Search by SSN**, **Update by SSN CleanTextFields** operations.
- ✓ SSN text field should be "read only" and **SSN** should be set in the code for each record, do not take from user.
- ✓ When program starts, read all records from random access file or text file and save each record in Employee []. (Employee array)
- Update both Employee [] and random access file or text file when **add** and **update** operations happen.
- ✓ **Add:** Select employee type from dropdown list. According to your selection, related text fields are enabled, unrelated text fields are disabled (Search/Update SSN text field always enabled). Enter employee information into enabled text fields, then press "add button". Your employee record will be written into file. Also add that record into your Employee[]
- **Search by SSN:** Enter SSN of employee whose information you want to reach, into "Search/Update SSN text field" . Press "Search by SSN button". Then all information related to the employee will be displayed, such as employee type, first name, last name, SSN, salary etc..
- **Update by SSN:** Enter SSN of employee whose information you want to update, into "Search/Update SSN text field". Enter other fields you want to update, then press "Update by SSN button**".** Employee record which is placed in file should be updated. Also, Employee[] should be updated.
- ✓ **Clean TextFields:** Cleans all text fields.

2- Submit a report (word or pdf) file that explains following questions with a cover page that contains your name and number. **In the report:**

- Indicate all project member's student No- Name surname in the report.
- Describe the problem including major steps, functions for solving the problem and input/output in your own words.
- Draw the UML diagram of your code.
- Describe how you test this program.