ÖZYEĞİN UNIVERSITY

CS 102
Object Oriented Programming

Company Example

(Modified from 2010 Pearson Education, Inc. Slides)

Reyyan Yeniterzi
reyyan.yeniterzi@ozyegin.edu.tr

Midterm

- Average: 48
- Success rate of questions
  - Q1: 40%
  - Q2: 51%
  - Q3: 48%
  - Q4: 50%

Quiz Question

- What are the differences between abstract classes and interfaces?

Quiz Question & Solution

- What are the differences between abstract classes and interfaces?

Abstract Class	Interface
At least one method needs to be abstract or the keyword abstract is used. (Some methods can be implemented)	All methods are abstract. (None of the methods have an implementation)
It can be **extended** by a class.	It can be **implemented** by a class.
A class can extend only 1 abstract class.	A class can implement several interfaces.
Any class can extend an abstract class.	Only an interface can extend another interface.
They can have attributes.	They cannot have attributes.

- Lets repeat classes over an example.

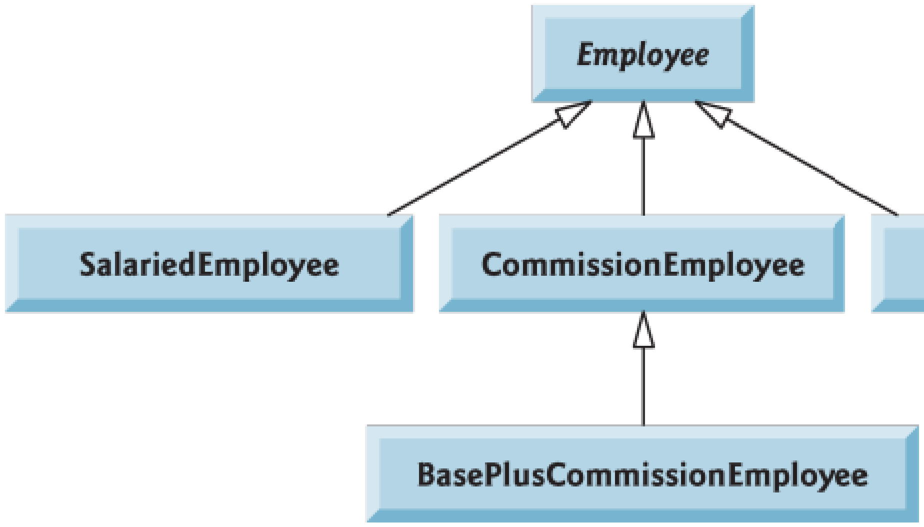
Example: Employee Inheritance

- A company pays its employees on a weekly basis and there are four types of employees:
  - **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-salaried commission employees** receive a base salary plus a percentage of their sales.

Example: Employee Inheritance

- For the current pay period, the company has decided to reward salaried-commission employees by adding 10% to their base salaries.
- The company wants to write a Java application that performs its payroll calculations polymorphically.
- What should be the inheritance hierarchy?

Example: Employee Inheritance



Employees

- Employees:
  - Attributes:
    - First name, last name and SSN
  - Behaviors:
    - They earn money (earnings function)

Employee Class

```
public class Employee {
    private String firstName, lastName;
    private int SSN;

    public Employee(String first, String last, int no) {
        firstName = first;
        lastName = last;
        SSN = no;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public int getSSN() {
        return SSN;
    }
    public String toString() {
        return firstName + " " + lastName + "\n"
                + "social security number: " + SSN;
    }
}
```

Employee Class

```
public class Employee {
    private String firstName, lastName;
    private int SSN;

    public Employee(String first, String last, int no) {
        firstName = first;
        lastName = last;
        SSN = no;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public int getSSN() {
        return SSN;
    }
    public String toString() {
        return firstName + " " + lastName + "\n"
                + "social security number: " + SSN;
    }
}
```

- Any thing missing?

Employee Class

```
public class Employee {
    private String firstName, lastName;
    private int SSN;

    public Employee(String first, String last, int no) {
        firstName = first;
        lastName = last;
        SSN = no;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public int getSSN() {
        return SSN;
    }
    public String toString() {
        return firstName + " " + lastName + "\n"
                + "social security number: " + SSN;
    }
}
```

- Any thing missing?
- How can we define the earnings method?

Employees

- A company pays its employees on a weekly basis and there are four types of employees:
  - **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-salaried commission employees** receive a base salary plus a percentage of their sales.

- We don’t have earnings description given for employee in default.
- All employees earn.
- All employees need to belong to one of the 4 types
- Therefore, …

- We don’t have earnings description given for employee in default.
- All employees earn.
- All employees need to belong to one of the 4 types
- Therefore, …
  - Earnings function should be abstract (incomplete)
  - Employee class should be abstract.

Employee Class

```
public abstract class Employee {
    private String firstName, lastName;
    private int SSN;

    public Employee(String first, String last, int no) {
        firstName = first;
        lastName = last;
        SSN = no;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public int getSSN() {
        return SSN;
    }
    public String toString() {
        return firstName + " " + lastName + "\n"
                + "social security number: " + SSN;
    }
    public abstract double earnings();
}
```

Salaried Employees

- **Salaried employees** are paid a fixed weekly salary regardless of the number of hours worked.
- One additional attribute
  - weeklysalary

Salaried Employees

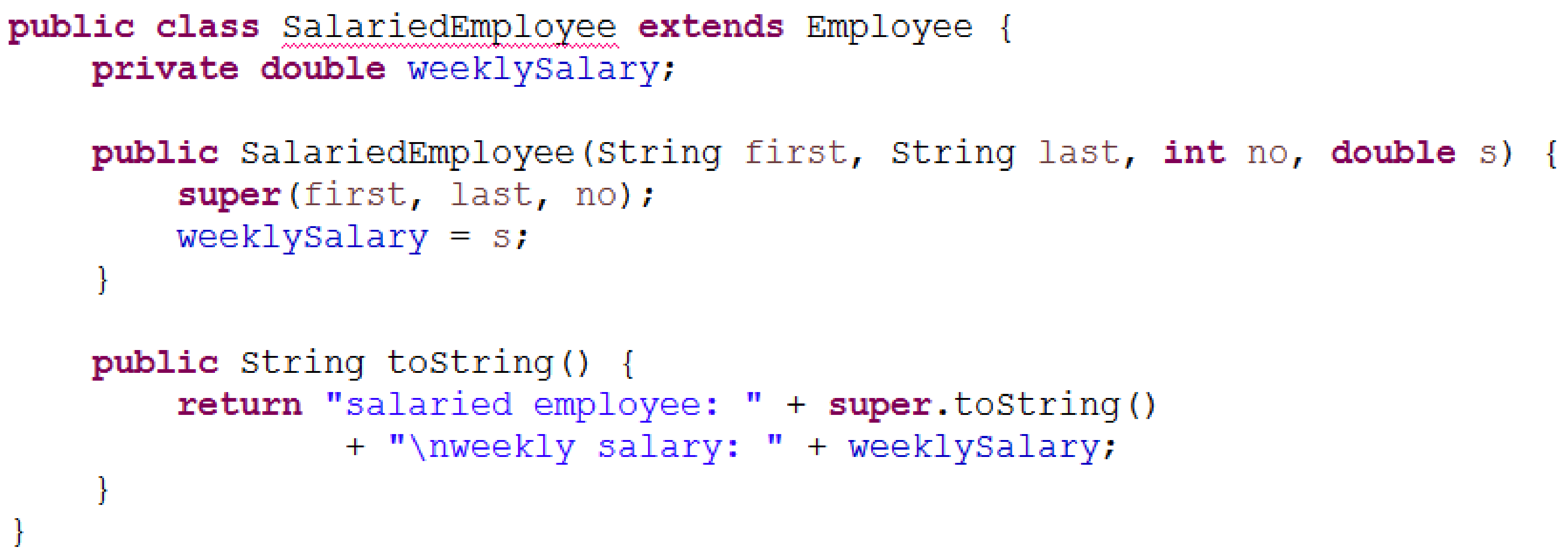
```
public class SalariedEmployee extends Employee {
    private double weeklySalary;

    public SalariedEmployee(String first, String last, int no, double s) {
        super(first, last, no);
        weeklySalary = s;
    }

    public String toString() {
        return "salaried employee: " + super.toString()
                + "\nweekly salary: " + weeklySalary;
    }
}
```

- What is wrong in here?

Salaried Employees



```
public class SalariedEmployee extends Employee {
    private double weeklySalary;

    public SalariedEmployee(String first, String last, int no, double s) {
        super(first, last, no);
        weeklySalary = s;
    }

    public String toString() {
        return "salaried employee: " + super.toString()
                + "\nweekly salary: " + weeklySalary;
    }
}
```

- What is wrong in here?
- Earnings function is missing.

Salaried Employees

```
public class SalariedEmployee extends Employee {
    private double weeklySalary;

    public SalariedEmployee(String first, String last, int no, double s) {
        super(first, last, no);
        weeklySalary = s;
    }
    public double earnings() {
        return weeklySalary;
    }
    public String toString() {
        return "salaried employee: " + super.toString()
                + "\nweekly salary: " + weeklySalary;
    }
}
```

- Is there a way to make sure that class cannot be extended?

Final Classes

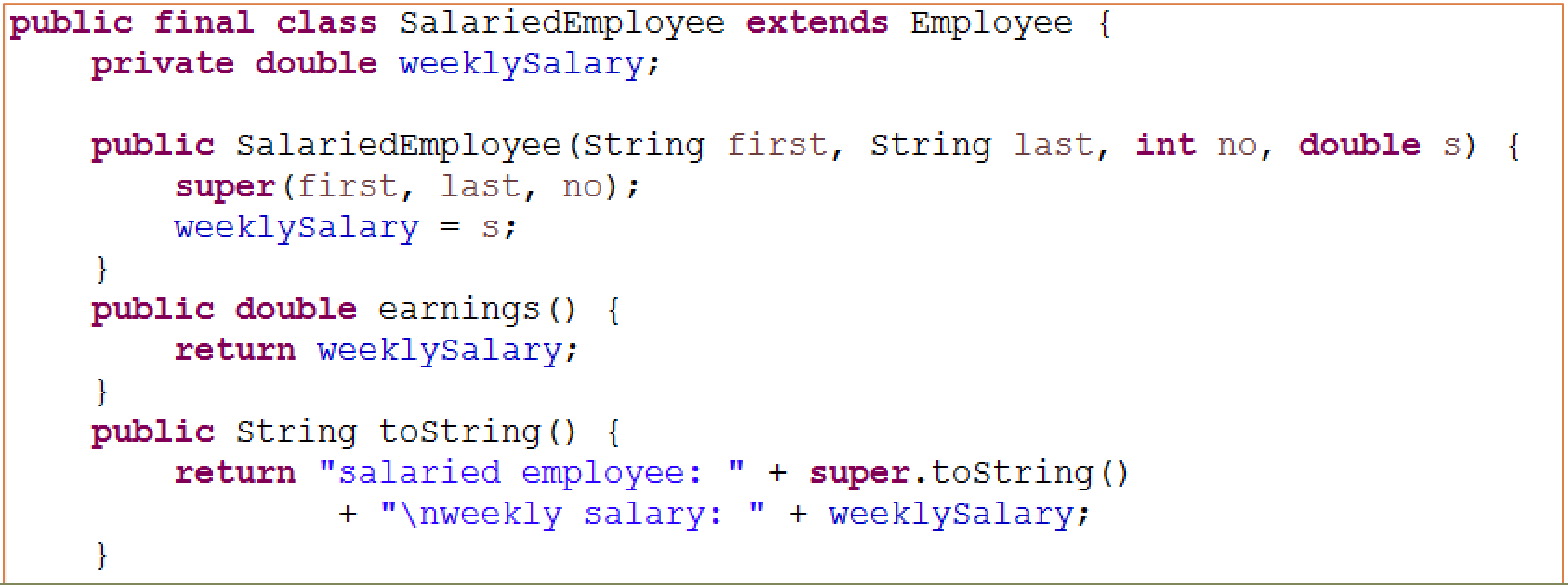
- By using the keyword final.

```
public final class SalariedEmployee extends Employee {
    private double weeklySalary;

    public SalariedEmployee(String first, String last, int no, double s) {
        super(first, last, no);
        weeklySalary = s;
    }
    public double earnings() {
        return weeklySalary;
    }
    public String toString() {
        return "salaried employee: " + super.toString()
                + "\nweekly salary: " + weeklySalary;
    }
}
```

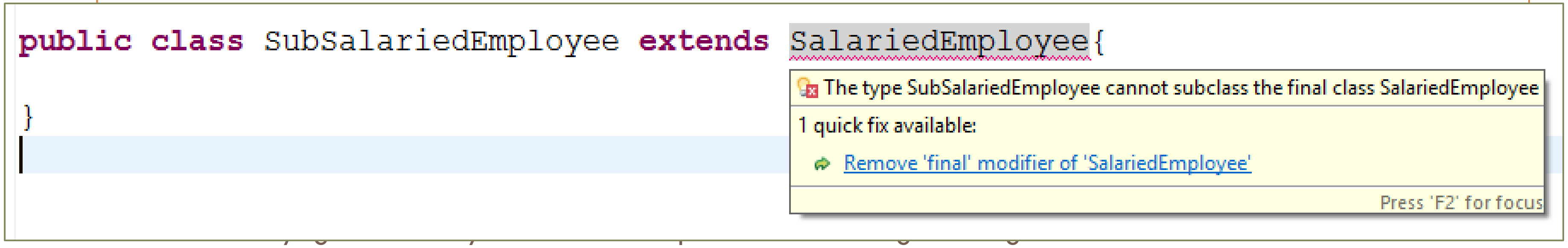
Final Classes

- By using the keyword final.



```
public final class SalariedEmployee extends Employee {
    private double weeklySalary;

    public SalariedEmployee(String first, String last, int no, double s) {
        super(first, last, no);
        weeklySalary = s;
    }
    public double earnings() {
        return weeklySalary;
    }
    public String toString() {
        return "salaried employee: " + super.toString()
                + "\nweekly salary: " + weeklySalary;
    }
```



```
public class SubSalariedEmployee extends SalariedEmployee{

}
```

The type SubSalariedEmployee cannot subclass the final class SalariedEmployee

1 quick fix available:

Remove 'final' modifier of 'SalariedEmployee'

Press 'F2' for focus

Final Classes

- Many standard java libraries are final for security and efficiency reasons.

Salaried Employees

```java
public final class SalariedEmployee extends Employee {
    private double weeklySalary;

    public SalariedEmployee(String first, String last, int no, double s) {
        super(first, last, no);
        weeklySalary = s;
    }
    public double earnings() {
        return weeklySalary;
    }
    public String toString() {
        return "salaried employee: " + super.toString()
                + "\nweekly salary: " + weeklySalary;
    }
}
```

Hourly Employees

- **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.
- Additional Attributes
  - ?

Hourly Employees

- **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.
- Additional Attributes
  - hours
  - salaryRate

Hourly Employees

- **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.
- Additional Attributes
  - hours
  - salaryRate
- Implementation of earnings method?

Hourly Employees

```
public final class HourlyEmployee extends Employee {
    private int hours;
    private double rate;

    public HourlyEmployee(String first, String last, int no, int h, double r) {
        super(first, last, no);
        hours = h;
        rate = r;
    }
    public double earnings() {
        if(hours <= 40) {
            return rate * hours;
        } else {
            return 40 * rate + (hours - 40) * rate * 1.5;
        }
    }
    public String toString() {
        return "hourly employee: " + super.toString()
                + "\nhourly wage: " + rate + " hours worked: " + hours;
    }
}
```

Comission Employees

- **Commission employees** are paid a percentage of their sales.
- Attributes:
  - ?

Comission Employees

- **Commission employees** are paid a percentage of their sales.
- Attributes:
  - comissionRate
  - grossSales
- Implementing earnings function?

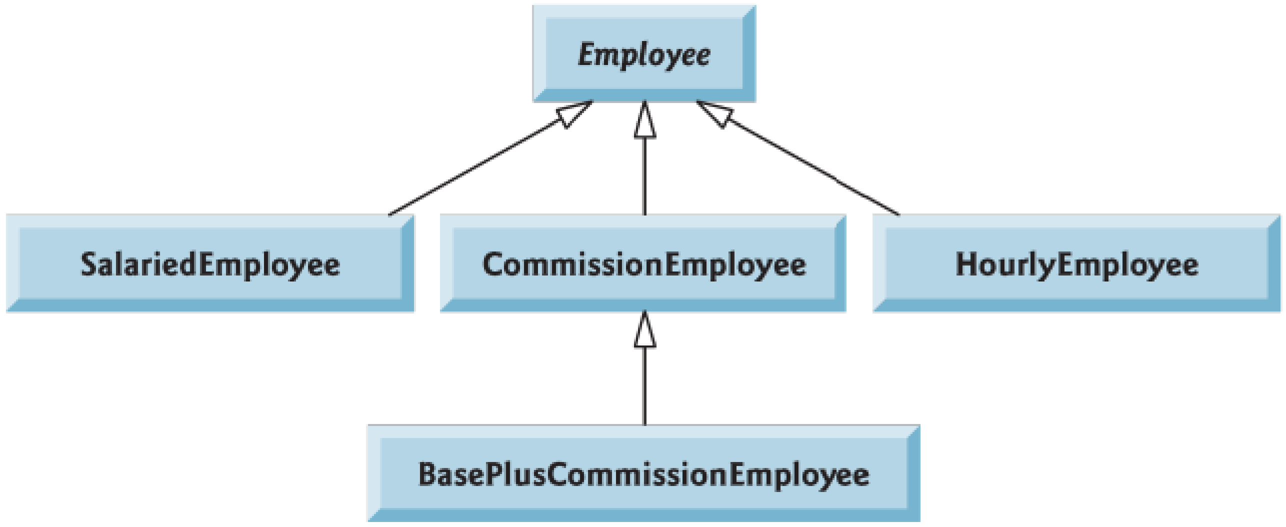
Comission Employees

```
public class CommissionEmployee extends Employee {
    private double commissionRate;
    private double grossSales;

    public CommissionEmployee(String first, String last,
            int no, double comm, double sales) {
        super(first, last, no);
        commissionRate = comm;
        grossSales = sales;
    }
    public double earnings() {
        return commissionRate * grossSales;
    }
    public String toString() {
        return "commission employee: " + super.toString()
                + "\ngross sales: " + grossSales
                + "\ncommission rate: " + commissionRate;
    }
}
```

Base-salaried Comission Employees

- **Base-salaried commission employees** receive a base salary plus a percentage of their sales.



Base-salaried Comission Employees

- **Base-salaried commission employees** receive a base salary plus a percentage of their sales.
- Additional attribute:
  - ?

Base-salaried Comission Employees

- **Base-salaried commission employees** receive a base salary plus a percentage of their sales.
- Additional attribute:
  - baseSalary

Base-salaried Comission Employees

```
public final class BasePlusCommissionEmployee extends CommissionEmployee {
    double baseSalary;

    public BasePlusCommissionEmployee(String first, String last, int no,
            double comm, double sales, double base) {
        super(first, last, no, comm, sales);
        baseSalary = base;
    }
    public double earnings() {
        return super.earnings() + baseSalary;
    }
    public String toString() {
        return "base salaried " + super.toString()
                + "\nbase salary: " + baseSalary;
    }
}
```

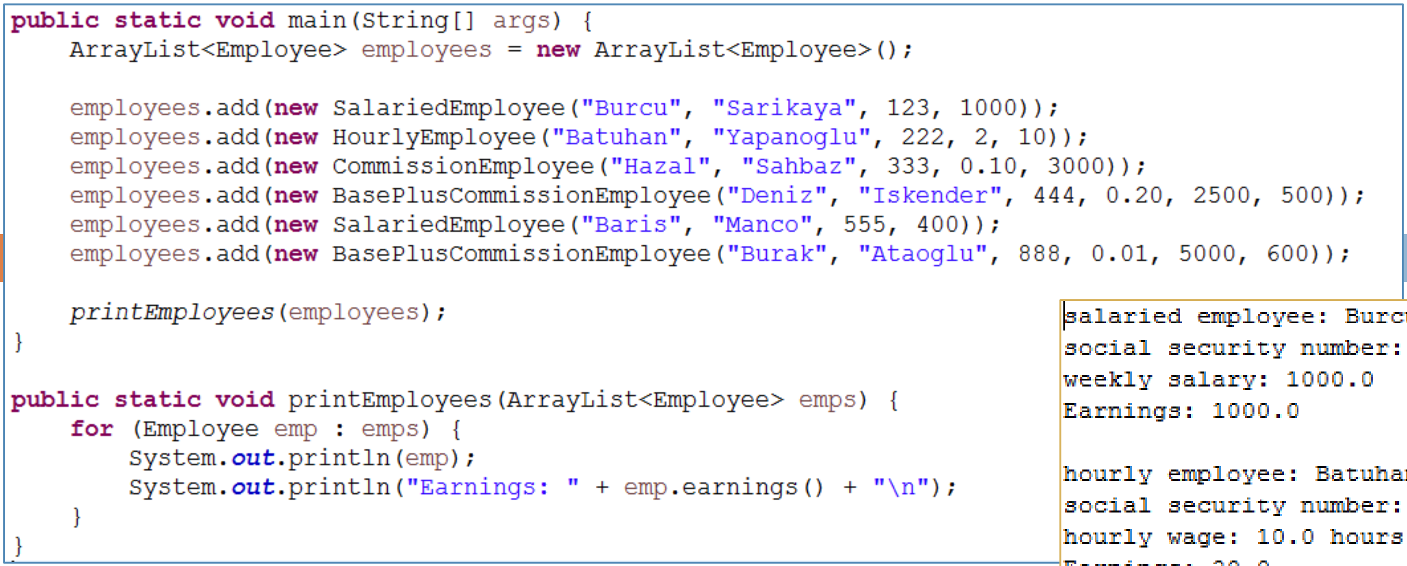
Test Class

```java
public static void main(String[] args) {
    ArrayList<Employee> employees = new ArrayList<Employee>();

    employees.add(new SalariedEmployee("Burcu", "Sarikaya", 123, 1000));
    employees.add(new HourlyEmployee("Batuhan", "Yapanoglu", 222, 2, 10));
    employees.add(new CommissionEmployee("Hazal", "Sahbaz", 333, 0.10, 3000));
    employees.add(new BasePlusCommissionEmployee("Deniz", "Iskender", 444, 0.20, 2500, 500));
    employees.add(new SalariedEmployee("Baris", "Manco", 555, 400));
    employees.add(new BasePlusCommissionEmployee("Burak", "Ataoglu", 888, 0.01, 5000, 600));

    printEmployees(employees);
}

public static void printEmployees(ArrayList<Employee> emps) {
    for (Employee emp : emps) {
        System.out.println(emp);
        System.out.println("Earnings: " + emp.earnings() + "\n");
    }
}
```

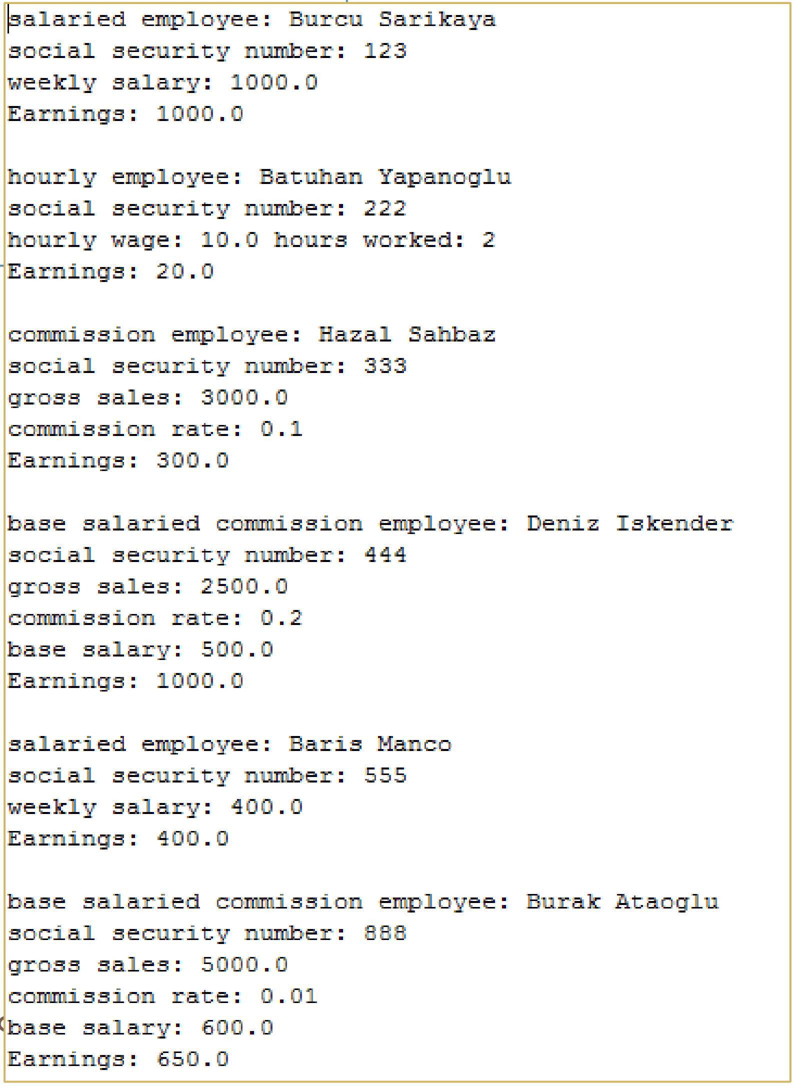


```java
public static void main(String[] args) {
    ArrayList<Employee> employees = new ArrayList<Employee>();

    employees.add(new SalariedEmployee("Burcu", "Sarikaya", 123, 1000));
    employees.add(new HourlyEmployee("Batuhan", "Yapanoglu", 222, 2, 10));
    employees.add(new CommissionEmployee("Hazal", "Sahbaz", 333, 0.10, 3000));
    employees.add(new BasePlusCommissionEmployee("Deniz", "Iskender", 444, 0.20, 2500, 500));
    employees.add(new SalariedEmployee("Baris", "Manco", 555, 400));
    employees.add(new BasePlusCommissionEmployee("Burak", "Ataoglu", 888, 0.01, 5000, 600));

    printEmployees(employees);
}

public static void printEmployees(ArrayList<Employee> emps) {
    for (Employee emp : emps) {
        System.out.println(emp);
        System.out.println("Earnings: " + emp.earnings() + "\n");
    }
}
```



```
salaried employee: Burcu Sarikaya
social security number: 123
weekly salary: 1000.0
Earnings: 1000.0

hourly employee: Batuhan Yapanoglu
social security number: 222
hourly wage: 10.0 hours worked: 2
Earnings: 20.0

commission employee: Hazal Sahbaz
social security number: 333
gross sales: 3000.0
commission rate: 0.1
Earnings: 300.0

base salaried commission employee: Deniz Iskender
social security number: 444
gross sales: 2500.0
commission rate: 0.2
base salary: 500.0
Earnings: 1000.0

salaried employee: Baris Manco
social security number: 555
weekly salary: 400.0
Earnings: 400.0

base salaried commission employee: Burak Ataoglu
social security number: 888
gross sales: 5000.0
commission rate: 0.01
base salary: 600.0
Earnings: 650.0
```

- Now we have the classes working correctly.
- Lets implement a solution for our problem.
  - For the current pay period, the company has decided to reward salaried-commission employees by adding 10% to their base salaries.
  - How should we implement?

- For the current pay period, the company has decided to reward salaried-commission employees by adding 10% to their base salaries.
- How should we implement?
  - Similar to print function, iterate over all employees?

```
public static void printEmployees(ArrayList<Employee> emps) {
    for (Employee emp : emps) {
        System.out.println(emp);
        System.out.println("Earnings: " + emp.earnings() + "\n");
    }
}
```

  - But we need to find the salaried-commission employees. How?

- For the current pay period, the company has decided to reward salaried-commission employees by adding 10% to their base salaries.
- How should we implement?
  - Similar to print function, iterate over all employees?

```
public static void printEmployees(ArrayList<Employee> emps) {
    for (Employee emp : emps) {
        System.out.println(emp);
        System.out.println("Earnings: " + emp.earnings() + "\n");
    }
}
```

  - But we need to find the salaried-commission employees. How?
    - by using **instanceof** operator

```
public static void promoteBasePlusCommissionEmployees(ArrayList<Employee> emps) {
    for(Employee emp: emps) {
        if(emp instanceof BasePlusCommissionEmployee) {
            BasePlusCommissionEmployee bemp = (BasePlusCommissionEmployee)emp;
            bemp.baseSalary = bemp.baseSalary * 1.10;
        }
    }
}
```

```
public static void promoteBasePlusCommissionEmployees(ArrayList<Employee> emps) {
    for(Employee emp: emps) {
        if(emp instanceof BasePlusCommissionEmployee) {
            BasePlusCommissionEmployee bemp = (BasePlusCommissionEmployee)emp;
            bemp.baseSalary = bemp.baseSalary * 1.10;
        }
    }
}
```

- We can do this because baseSalary is package-private.
- We can make is private and use get and set methods.

Base-salaried Comission Employees

```
public final class BasePlusCommissionEmployee extends CommissionEmployee {
    private double baseSalary;

    public BasePlusCommissionEmployee(String first, String last, int no,
            double comm, double sales, double base) {
        super(first, last, no, comm, sales);
        baseSalary = base;
    }
    public double earnings() {
        return super.earnings() + baseSalary;
    }
    public String toString() {
        return "base salaried " + super.toString()
                + "\nbase salary: " + baseSalary;
    }
    public double getBaseSalary() {
        return baseSalary;
    }
    public void setBaseSalary(double salary) {
        baseSalary = salary;
    }
}
```

- promoteBasePlusCommissionEmployees method with get and set methods.

```java
public static void promoteBasePlusCommissionEmployees(ArrayList<Employee> emps) {
    for(Employee emp: emps) {
        if(emp instanceof BasePlusCommissionEmployee) {
            BasePlusCommissionEmployee bemp = (BasePlusCommissionEmployee)emp;
            //bemp.baseSalary = bemp.baseSalary * 1.10;
            bemp.setBaseSalary(bemp.getBaseSalary() * 1.10);
        }
    }
}
```

Next example

- Employees and invoices are two things that a company pays for.
- Build an application that can determine payments for employees and invoices.

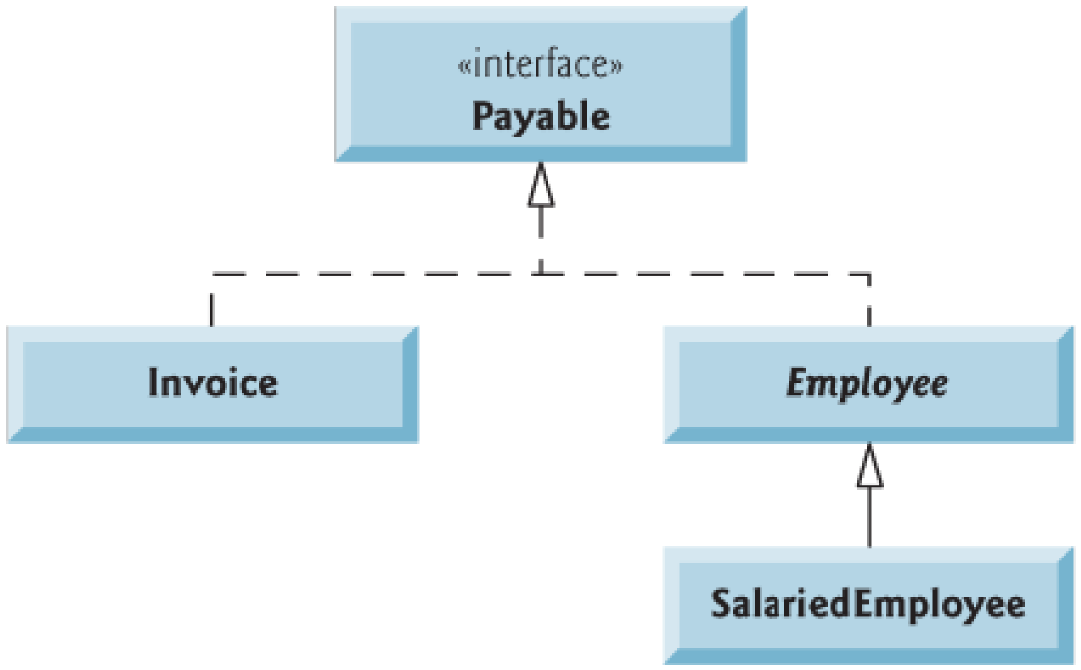
Next example

- We need to calculate the payment amount.
  - getPayment amount needs to be calculated for both invoices and employees.
  - invoice and employee are two very different classes, they don't share any common attributes but they need to call the same method.
  - any idea how?

Next example

- We need to calculate the payment amount.
  - getPayment amount needs to be calculated for both invoices and employees.
  - invoice and employee are two very different classes, they don’t share any common attribute but they need to call the same method.
  - any idea how?
    - Think of a **payable** interface with **getPayment()** method.
    - **invoice** and **employee** classes implement the **payable** interface.

Class diagram



- We distinguish an interface from other classes by placing a `<<interface>>` above the interface name. (See the UML (Unified Modeling Language) slides)

Payable Interface

- Interface methods are implicitly public and abstract

```
public interface Payable {

    double getPayableAmount();
}
```

Invoice Class

- Implements payable interface
  - getPayableAmount();
- Attributes
  - ?

Invoice Class

- Implements payable interface
  - getPayableAmount();
- Attributes
  - pricePerItem
  - quantity

Invoice Class

```
public class Invoice implements Payable {
    private int quantity;
    private double pricePerItem;

    public Invoice(int q, double p) {
        quantity = q;
        pricePerItem = p;
    }
    public double getPayableAmount() {
        return quantity * pricePerItem;
    }
    public String toString() {
        return super.toString() + " quantity = " + quantity
                + " price per item = " + pricePerItem
                + " total price   = " + getPayableAmount();
    }
}
```

Employee Class

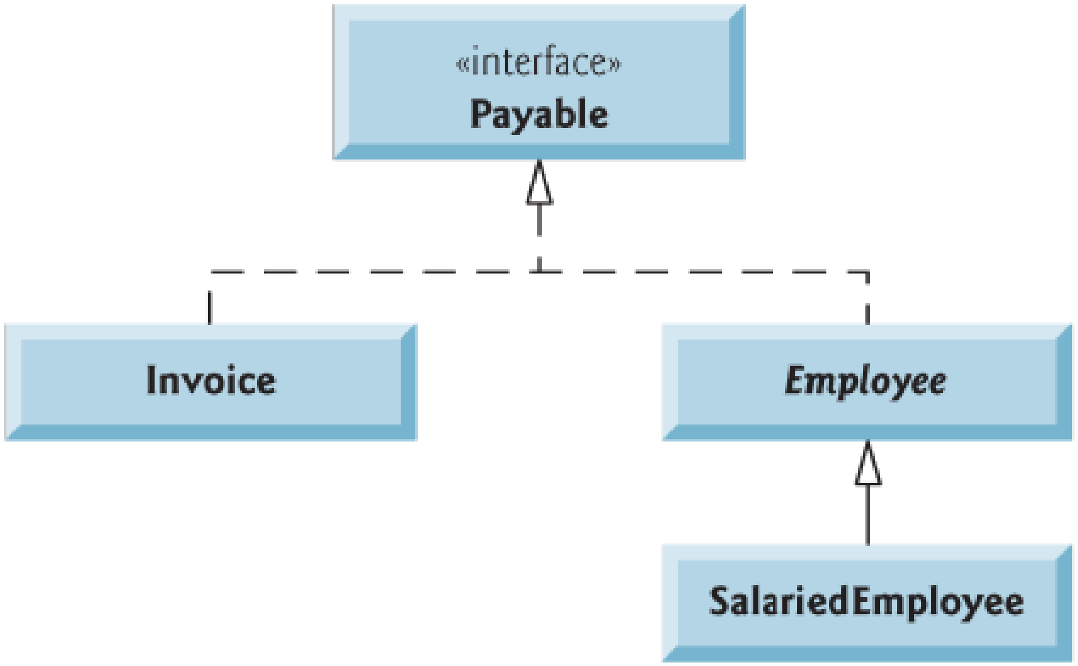
- Employee Class will be modified.
- We no longer have `earnings()` method, we will implement `getPayableAmount()` method of Payable.

Employee Class

```
public abstract class Employee implements Payable {
    private String firstName, lastName;
    private int SSN;

    public Employee(String first, String last, int no) {
        firstName = first;
        lastName = last;
        SSN = no;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public int getSSN() {
        return SSN;
    }
    public String toString() {
        return firstName + " " + lastName + "\n"
                + "social security number: " + SSN;
    }
}
```

- Still abstract.
- Why?


«interface»
Payable
Invoice
Employee
SalariedEmployee

SalariedEmployee Class

```
public class SalariedEmployee extends Employee {
    private double weeklySalary;

    public SalariedEmployee(String first, String last, int no, double s) {
        super(first, last, no);
        weeklySalary = s;
    }
    public double getPayableAmount() {
        return weeklySalary;
    }
    public String toString() {
        return "salaried employee: " + super.toString()
                + "\nweekly salary: " + weeklySalary;
    }

}
```

- SalariedEmployee Class is concrete now.

Testing

```
public class Company {
    public static void main(String[] args) {
        Payable[] payables = new Payable[2];
        payables[0] = new Invoice(10, 3.5);
        payables[1] = new SalariedEmployee("Mary", "Jane", 1234, 1000);

        double total = 0;
        for(Payable p : payables) {
            System.out.println(p);
            total += p.getPayableAmount();
        }
        System.out.println("Total = "+ total);
    }
}
```

- Payable can be the reference to both Invoice and Employee objects.

Testing

```
public class Company {
    public static void main(String[] args) {
        Payable[] payables = new Payable[2];
        payables[0] = new Invoice(10, 3.5);
        payables[1] = new SalariedEmployee("Mary", "Jane", 1234, 1000);

        double total = 0;
        for(Payable p : payables) {
            System.out.println(p);
            total += p.getPayableAmount();
        }
        System.out.println("Total = "+ total);
    }
}
```

```
company.Invoice@2a139a55 Quantity = 10 price per item = 3.5 total price  = 35.0
salaried employee: Mary Jane
social security number: 1234
weekly salary: 1000.0
Total = 1035.0
```

Any Questions ?