Polymorphic Payroll Calculation Using Inheritance

Implement a payroll system for a company using object-oriented programming principles, particularly **inheritance**, **abstraction**, and **polymorphism**.

You are required to develop a **Java application** that models a company's payroll system. The company employs four types of employees:

1. **SalariedEmployee** – paid a fixed weekly salary.
2. **HourlyEmployee** – paid by the hour, with overtime (1.5x) for hours worked over 40.
3. **CommissionEmployee** – paid a percentage of their sales.
4. **BasePlusCommissionEmployee** – paid a base salary plus a percentage of their sales. For the current pay period, their base salary must be increased by 10% as a reward.

Instructions:

1. Create an **abstract class** `Employee` that includes the following:
   - Instance variables: first name, last name, and social security number.
   - Abstract method `double earnings()`.
   - Concrete method `String toString()`.
2. Implement the following subclasses:
   - `SalariedEmployee` with a weekly salary.
   - `HourlyEmployee` with hourly wage and hours worked.
   - `CommissionEmployee` with gross sales and commission rate.
   - `BasePlusCommissionEmployee` (inherits from `CommissionEmployee`) with an additional base salary.
3. Override the `earnings()` and `toString()` methods in each subclass to reflect the appropriate logic for earnings and string representation.
4. In your **main method**:
   - Create an array of `Employee` references and initialize it with objects of each subclass.
   - Increase the **base salary** of all `BasePlusCommissionEmployee` objects by **10%**.
   - Iterate through the array and for each employee:
     - Display their information using `toString()`.
     - Display their weekly earnings using the `earnings()` method.

You must demonstrate polymorphic behavior by invoking the `earnings()` and `toString()` methods on `Employee` references, ensuring the correct subclass versions are executed.

The below UML class diagram shows the inheritance hierarchy for our polymorphic employee-payroll application. Abstract class name Employee is italicized a convention of the UML.