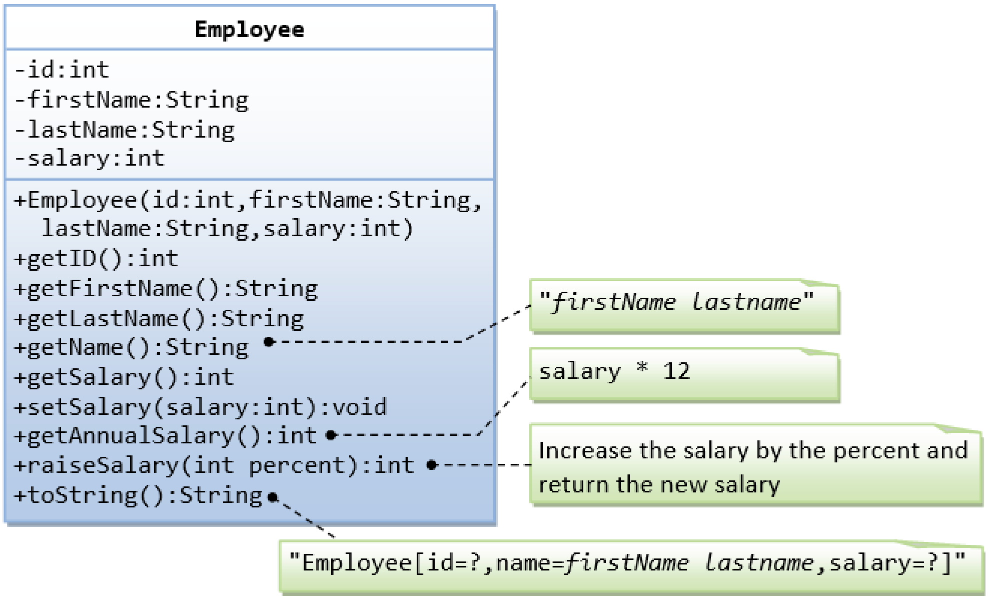
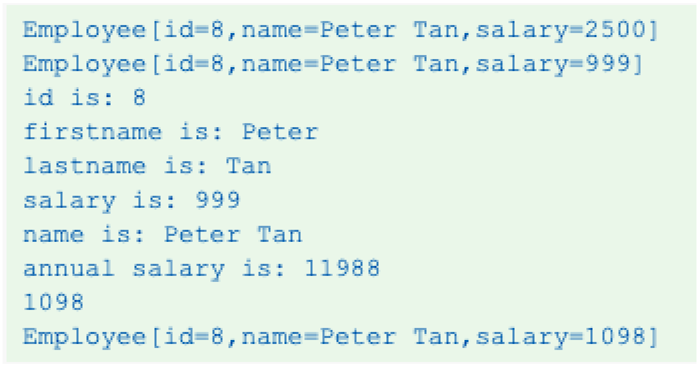
Practical May 19

1) A class called `Employee`, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method `raiseSalary(percent)` increases the salary by the given percentage. Write the `Employee` class and a test driver class `EmployeeDemo`.


Employee
-id:int
-firstName:String
-lastName:String
-salary:int
+Employee(id:int,firstName:String,
lastName:String,salary:int)
+getID():int
+getFirstName():String
+getLastName():String
+getName():String
+getSalary():int
+setSalary(salary:int):void
+getAnnualSalary():int
+raiseSalary(int percent):int
+toString():String
"firstName lastname"
salary * 12
Increase the salary by the percent and return the new salary
"Employee[id=?,name=firstName lastname,salary=?]"

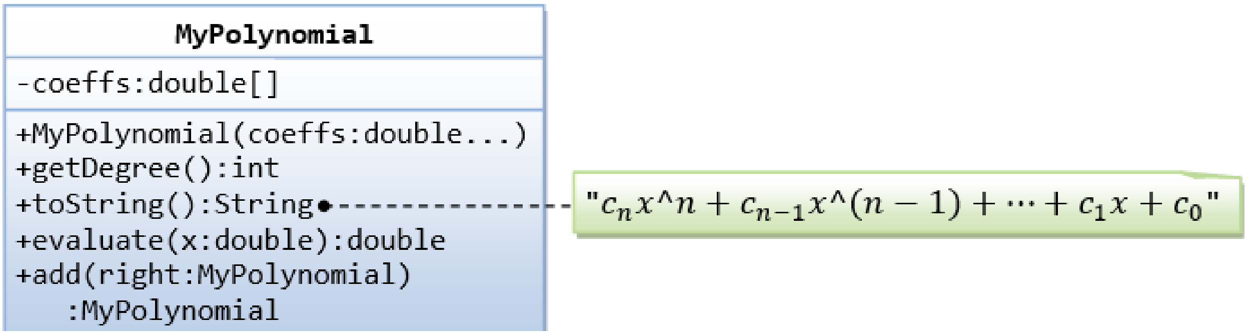
Expected Output:



```
Employee[id=8,name=Peter Tan,salary=2500]
Employee[id=8,name=Peter Tan,salary=999]
id is: 8
firstname is: Peter
lastname is: Tan
salary is: 999
name is: Peter Tan
annual salary is: 11988
1098
Employee[id=8,name=Peter Tan,salary=1098]
```

2) A class called `MyPolynomial`, which models polynomials of degree-n (see equation), is designed as shown in the class diagram.

$$c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0.$$



```
MyPolynomial
-coeffs:double[]
+MyPolynomial(coeffs:double...)
+getDegree():int
+toString():String
+evaluate(x:double):double
+add(right:MyPolynomial)
   :MyPolynomial
```

$"c_n x\hat{}n + c_{n-1} x\hat{}(n-1) + \cdots + c_1 x + c_0"$

It contains:

- An instance variable named `coeffs`, which stores the coefficients of the *n*-degree polynomial in a `double` array of size `n+1`, where c_0 is kept at index 0.
- A constructor `MyPolynomial(coeffs:double[])` that takes a double array to initialize the coefficients.
  ```
  double coefficients[] = {1.2, 3.4, 5.6, 7.8};
  MyPolynomial p1 = new MyPolynomial(coefficients);
  ```
- A method `getDegree()` that returns the degree of this polynomial.
- A method `toString()` that returns "c_nx^n+c_{n-1}x^(n-1)+...+c_1x+c_0".
- A method `evaluate(double x)` evaluates the polynomial for the given `x`, by substituting the given `x` into the polynomial expression.
- Method `add()` that adds this polynomial with the given `MyPolynomial` instance `another`, and returns `this` instance that contains the result.

Write the `MyPolynomial` class. Also write a test driver (called `MyPolynomialDemo`) to test all the methods defined in the class.