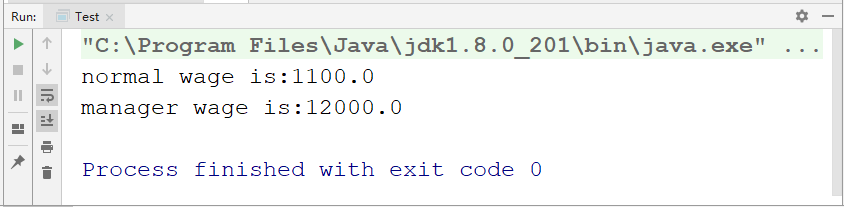
【案例4-6】经理与员工工资案例（利用多态实现）

**【案例介绍】**

1. **案例描述**

某公司的人员分为员工和经理两种，但经理也属于员工中的一种，公司的人员都有自己的姓名和地址，员工和经理都有自己的工号、工资、工龄等属性，但经理不同员工的是，经理有自己在公司对应的级别。假设每次给员工涨工资一次能涨10%，经理能涨20%。本案例要求利用多态实现给员工和经理涨工资。

1. **运行结果**



**【案例目标】**

* 学会利用继承特性实现“经理与员工工资案例”的实现思路。
* 独立完成“经理与员工工资案例”的源代码编写、编译及运行。
* 掌握多态的概念及其应用。
* 掌握方法的重载的概念及其应用。

**【案例分析】**

（1）创建父类Person类，在Person类中定义name和address属性，并定义该类的构造方法。

（2）创建抽象类Employee类并继承Person类，创建构造方法，在构造方法中调用父类的构造方法。在Employee类中定义员工的ID、工资wage、年龄age等属性；在类中定义涨工资的抽象方法add()，通过对职位的判断来给员工或经理涨工资。

（3）创建子类Manager类并继承Employee类；创建构造方法，在构造方法中调用父类的构造方法；由于经理有两种身份，既是Employee又是Manager，所以Manager类继承Employee类，在Manager类中定义等级属性level，并给出level的getter和setter方法；实现Employee类的add()抽象方法。

（4）创建测试类，对Manager进行实例化，传入参数，调用涨薪方法，传入级别level参数，根据级别level输出涨薪工资。

**【案例实现】**

Person.java

1. public class Person {
2. private String name = "";
3. private String address = "";
4. //定义构造方法
5. public Person(String name, String address){
6. this.name = name;
7. this.address = address;
8. }
9. }

上述代码中，创建了一个父类Person类，在Person类中定义了name和address属性以及Person类的构造方法。

Employee.java

1. public abstract class Employee extends Person {
2. private String ID = "";
3. private double wage = 0;
4. private int age = 0;
5. public Employee(String name, String address, String ID, double
6. wage, int age){
7. super(name, address);
8. this.ID = ID;
9. this.wage = wage;
10. this.age = age;
11. }
12. //定义抽象方法
13. public abstract void add(String position);
14. //设置get/set方法
15. public double getWage() {
16. return wage;
17. }
18. public void setWage(double wage) {
19. this.wage = wage;
20. }
21. }

上述代码中，创建了一个Employee类并继承了Person类，在Employee类中，定义了Employee类的构造方法，并在构造方法中调用了父类Person的构造方法；定义了员工的name和wage和age属性并提供了setter和getter方法；定义了抽象方法add()。

Manager.java

1. public class Manager extends Employee{
2. private String level = "";
3. public Manager(String name, String address, String ID, double wage,
4. int age, String level){
5. super(name, address, ID, wage, age);
6. this.level = level;
7. }
8. //实现抽象方法
9. public void add(){
10. double wage = super.getWage();
11. super.setWage(wage\*1.1);
12. }
13. public void add(String position){
14. double wage = super.getWage();
15. super.setWage(wage\*1.2);
16. }
17. public String getLevel() {
18. return level;
19. }
20. public void setLevel(String level) {
21. this.level = level;
22. }
23. }

上述代码中，创建了一个Manager类并继承了Employee类，在Manager类中，定义了Manager类的构造方法，并在构造方法中调用了父类Employee的构造方法；定义了员工的级别level属性并提供了setter和getter方法；实现了Employee类的抽象方法add()，并对add()方法进行了重载。

Test.java

1. public class Test {
2. public static void main(String[] args) {
3. Manager normal = new Manager("wsl", "jit", "12", 1000, 2, "1");
4. Manager manager = new Manager("ctl", "jitt", "123", 10000, 10,
5. "0");
6. normal.add();
7. manager.add(manager.getLevel());
8. System.out.println("normal wage is:"+normal.getWage());
9. System.out.println("manager wage is:"+manager.getWage());
10. }
11. }

上述代码中，创建了一个测试类Test类，在类中对Manager进行了实例化并传参。类中通过调用无参的add()方法来获取普通员工的涨薪，调用有参的add(manager.getLevel())方法来获取经理的涨薪。