Java的三大基本特征是什么？ 尝试编写一些示例代码与理论相验证

封装，继承，多态

```
package yichang;

class Person {

public String name;

public Person(String name) {
    this.name=name;
}

public void eat(){
    System.out.println("吃饭");
}
}
class Student extends Person{
//继承，Student 类继承自 Person 类，继承了 Person 类的 name 属性和 eat 方法
private int grades;
//封装，外部不能直接访问 grades 属性，只能通过构造方法来设置其初始值

public Student(String name, int grades) {
    super(name);
    this.grades = grades;
}

@Override
public void eat() {
    System.out.println("吃铁板烧");
}
//多态意味着同一个方法调用可以依据对象的不同类型呈现出不同的行为，Student 类重写了 Person 类的 eat 方法
}

public class jiancha {
public static void main(String[] args) {
    Student student=new Student("张三",59);
    student.eat();
}
}
```

抽象类

取决于有没有抽象方法，只要这个类有抽象方法，就必须要加abstra
为什么要有抽象类：在写其他类的父类时，有一个方法只有申明，没有方法体，必须要在子类中重写，就有了抽象方法，也就有了抽象类
所以说抽象类就是一个特殊的父类，强制子类来实现自己的方法
在上面多态的例子中，Person类中的eat方法如果改写为抽象方法，那么就没有
System.out.println("吃饭");
这个方法体，并且子类student必须要重写这个方法。

接口

和抽象类比较相似，一个类只能继承一个抽象类，但是可以实现多个接口，二者的侧重点不同，抽象类更像是表示子父辈之间的关系，而接口代表了can-do的关系

```
interface Perarea {
double get_area();
double get_perimeter();
}

class Chang implements Perarea {
private double length;
private double width;

public Chang(double length, double width) {
    this.length = length;
    this.width = width;
}

@Override
public double get_area() {
    return length * width;
}

@Override
public double get_perimeter() {
    return 2 * (length + width);
}
}

class Circle implements Perarea {
private double radius;

public Circle(double radius) {
    this.radius = radius;
}

@Override
public double get_area() {
    return Math.PI * radius * radius;
}

@Override
public double get_perimeter() {
```

```
    return 2 * Math.PI * radius;
}
}

public class Main {
public static void main(String[] args) {

    Perarea Chang = new Chang(10, 5);

    Perarea circle = new Circle(5);


    System.out.println("长方形的面积: " + Chang.get_area());
    System.out.println("长方形的周长: " + Chang.get_perimeter());


    System.out.println("圆形的面积: " + circle.get_area());
    System.out.println("圆形的周长: " + circle.get_perimeter());
}
}
```

.

```
public class Main {
public static void main(String[] args) {
    PlainRect plainRect = new PlainRect(10, 10, 20, 10);

    System.out.println("矩形的面积是: " + plainRect.area());
    System.out.println("矩形的周长是: " + plainRect.perimeter());

    boolean isInside = plainRect.isInside(25.5, 13);
    if (isInside) {
        System.out.println("点 (25.5, 13) 在矩形内。");
    } else {
        System.out.println("点 (25.5, 13) 不在矩形内。");
    }
}
}
class Rect{
public double width;
public double heigth;

public Rect(double heigth, double width) {
    this.heigth = heigth;
    this.width = width;
}

public Rect(){
    this.width=10;
    this.heigth=10;
}

public double area(){
    return width*heigth;
}

public double perimeter(){
    return 2*(width+heigth);
}
}

class PlainRect extends Rect {
private double startX;
private double startY;

public PlainRect(double startX, double startY, double width, double height) {
    super(width, height);
    this.startX = startX;
    this.startY = startY;
}

public PlainRect() {
    super(0, 0);
    this.startX = 0;
    this.startY = 0;
}

public boolean isInside(double x, double y) {
    return x >= startX && x <= startX + width && y >= startY && y <= startY + height;
}
}
```