抽象类

public abstract class Person{

public abstract String getDescription();

}

（1） 抽象类中的抽象方法只是声明，不给出方法的具体实现；

（2） 有抽象方法的类必须被声明为抽象类，抽象类中不一定包含抽象方法；

（3） 构造方法和静态方法不能声明为抽象方法；

（4） 抽象类不能实例化（不能new），但可以定义抽象类的对象变量（只能引用非抽象子类对象）

例：new Person() //Person为抽象类，该语句不合法

Person p=new Student();// Student为非抽象子类

（5） 抽象类必须被继承，抽象方法必须在子类中被重写，或再次声明为抽象类。

final关键字

 final变量值不能被改变（成员变量、局部变量（形参））

 final方法不能被重写

 final类不能被继承

子类构造器

例：public Manager(String name, double salary, int year, int month, int day)

{

super(name, salary, year, month, day);

Bonus=0;

} //子类中的构造函数

 子类的构造过程中必须调用父类的构造方法。

 如果子类构造器没有显式调用，则默认调用无参数构造器。

 如果没有显示调用父类构造方法，又没有无参数构造器，则编译错误。

 可以使用super调用父类构造方法，也可以使用this调用本类其他构造方法。如果调用super，必须写在子类构造方法的第一行。

this关键字

 一般出现在方法里面，指当前对象。可以看做是一个变量，值是当前对象的引用。

 可以处理方法中成员变量和参数重名的情况，如构造方法。

 一个构造器调用同一个类的另一个构造器：（this的第二种用法）

多态

一个对象变量可以指示多种实际类型的现象称为多态（polymorphism），在运行时能自动选择调用哪个方法的现象称为动态绑定（dynamic binding）。

父类变量可以引用任何一个子类对象，不能将一个超类的引用赋给子类变量。

子类数组的引用可以转换成超类数组的引用，反之不行。（见《核心》p155）

在执行期间（分配内存了）判断所引用对象的实际类型，根据实际的类型调用相应的方法。调用方法时，只要方法重写了，就按照实际new的对象来调用。提高可扩展性。

三个必要条件：

 要有继承

 要有重写

 父类引用指向子类对象

接口（实现多继承）

（1）接口不是类（本质上是特殊的抽象类），只包含常量和方法的定义，没有变量和方法的实现（不能用new实例化接口）：

 接口中只能定义抽象方法，这些方法只能是public，不声明默认为public；

例：public interface Comparable{

(public) int compareTo(Object other);

}

（2）负责实现接口的类完成方法实现以及实例域：

 将类声明为实现给定的接口（关键字implements）

 对接口中的所有方法进行定义

 在实现接口时必须把方法声明为public

例：class Employee implements Comparable{

public int compareTo(Object otherObject)

{

Employee other = (Employee) otherObject;

return Double.compare(salary, other.salary);

}

}

（3）接口的特性

 不能构造接口的对象，可以声明接口的变量，接口变量必须引用实现了接口的类对象

例：x=new Comparable(…); //ERROR

Comparable x;

x=new Employee(…);

 可以用instancof检查一个对象是否实现了某个特定的接口

if(anObject instanceof Comparable){}

 接口之间可以相互继承：Public interface Powered extends Moveable，都要实现

 接口可以多重实现：class Employee implements Cloneable,Compareble

 不能包含实例域或静态方法，但可以包含常量。常量必须定义为public static final，避免出现多个接口具有多个成员变量的问题。

Public interface Powered extends Moveable{

Double SPEED\_LIMIT = 95; // 不写自动设置为public static final

}

接口可以继承其他的接口，并添加新的属性和抽象方法

和继承关系类似，接口与实现类之间存在多态性：父类引用指向子类对象。

Singer s1=new student(“li”);//singer为接口，student为实现类，s1只能引用接口内方法。可以强制转换：Protectable p=(Protectable) s1;//这样p就可以看到protectable里的方法

多个无关类可实现同一个接口，一个类可以实现多个无关的接口。

抽象类和接口有什么异同

 抽象类只能扩展于一个类，每个类可以实现多个接口（1对多）

class Employee extends Person implements Comparable//Em为Person的子类

 Java不支持多重继承，接口允许一个类有多个超类