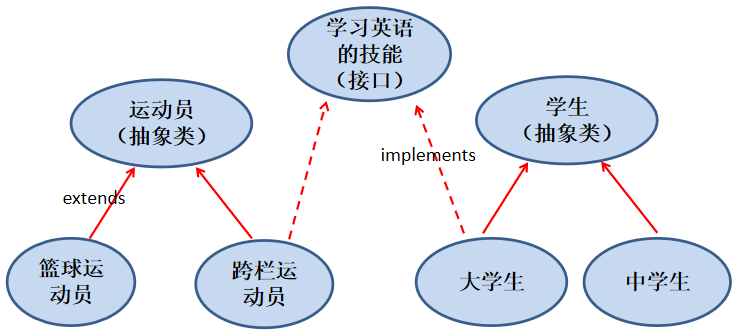
* **有时必须从几个类中派生出一个子类，继承他们所有的属性和方法。但是，Java不支持多重继承。有了接口，就可以得到多重继承的效果。**
* **接口（interface）是抽象方法和常量值的定义的集合。**
* **从本质上讲，接口是一种特殊的抽象类，这种抽象类中只包含常量和方法的定义，而没有变量和方法的实现。**
* **实现接口类：**
* class SubClass **implements** InterfaceA {}
* **一个类可以实现多个接口，接口也可以继承其它接口。**



* **接口的特点：**
* 用interface来定义。
* 接口中的所有**成员变量都默认是由public static final修饰的**。
* 接口中的所有**方法都默认是由public abstract修饰的。**
* **接口没有构造器**。
* 接口采用**多继承机制**。
* **接口定义举例：**

public interface Runner {

public static final int ID = 1;

public abstract void start();

public abstract void run();

public abstract void stop();

}

* **实现接口的类中必须提供接口中所有方法的具体实现内容，方可实例化。否则，仍为抽象类。**
* **接口的主要用途就是被实现类实现。（面向接口编程）**
* **与继承关系相似，接口与实现类之间存在多态性**

public class TestInterface1 {

public static void main(String[] args) {  
 Duck duck = new Duck();  
 TestInterface1.test1(duck);  
 TestInterface1.test2(duck);  
 TestInterface1.test3(duck);  
 }  
  
 /\*\*  
 \* Runner r = new Duck();  
 \*/  
 public static void test1(Runner r) {  
 r.run();  
 }  
  
 /\*\*  
 \* Swimmer s = new Duck();  
 \*/  
 public static void test2(Swimmer s) {  
 s.swim();  
 }  
  
 /\*\*  
 \* Flier f = new Duck();  
 \*/  
 public static void test3(Flier f) {  
 f.fly();  
 }  
}  
  
interface Runner {  
 public abstract void run();  
}  
  
interface Swimmer {  
 void swim();  
}  
  
interface Flier {  
 void fly();  
}  
  
class Duck implements Runner, Swimmer, Flier {  
  
 @Override  
 public void run() {  
 System.***out***.println("丑小鸭也可以变成白天鹅");  
 }  
 @Override  
 public void swim() {  
 System.***out***.println("红掌拨清波");  
 }  
 @Override  
 public void fly() {  
 System.***out***.println("鸭子屁股扭扭的走路");  
 }  
}

* **定义Java类的语法格式：先写extends，后写implements。**
* **接口也可以继承另一个接口，使用extends关键字。**
* **接口用法总结：**
* 通过接口可以实现不同类的相同行为，而不需要考虑这些类之间的层次关系。
* 通过接口可以指明多个需要实现的方法，一般用于定义对象的扩张功能。
* 接口主要用来**定义规范**。解除耦合关系。