我们可以说这个世界处处都是图，却不可以说这个世界处处都是树。（树是图的特例）

第 6 章 接口与实现

6.1 接口（C++无）

1. 接口声明

```
接口头 {
    接口体
}
```

2. 接口头和接口体

（1）接口头的格式为“interface 接口名”。
（2）接口体包含常量声明和抽象方法声明（不是定义）。
（3）常量声明开头的“public static final”可以省略。
（4）抽象方法声明开头的“public abstract”可以省略。

p.146 代码 1

```
interface Printable {
    public static final int MAX = 100;  //等价写法：int MAX = 100;
    public abstract void add();         //等价写法：void add();
    public abstract float sum(float x ,float y);
}
```

6.2 实现接口

1. 用类实现接口

（1）类头的格式为“class 类名 implements 接口名 1, 接口名 2, …, 接口名 n”。
（2）子类头的格式为“class 子类名 extends 父类名 implements 接口名 1, 接口名 2, …, 接口名 n”。

2. 重写接口中的方法

（1）用抽象类实现接口时，可以只重写接口的部分方法。
（2）用非抽象类实现接口时，需要重写接口的所有方法。
（3）重写方法的开头需要有“public”。

p.146 代码 4

```
public interface Computable {
    int MAX = 46;
    int f(int x);
}
```

pp.146-147 代码 5

```
public class China implements Computable {  //China 类实现 Computable 接口
   int number;
   public int f(int x) { //不要忘记 public 关键字
      int sum = 0;
      for(int i=1;i<=x;i++) {
         sum = sum+i;
      }
      return sum;
   }
}
```

p. 147 代码 2

```
public class Japan implements Computable { //Japan 类实现 Computable 接口
   int number;
   public int f(int x) {
      return MAX+x;  //直接使用接口中的常量
   }
}
```

p. 147 代码 3

```
public class Example6_1 {
   public static void main(String args[]) {
      China zhang;
      Japan henlu;
      zhang = new China();
      henlu = new Japan();
      zhang.number = 32+Computable.MAX;  //用接口名访问接口的常量
      henlu.number = 14+Computable.MAX;
      System.out.println("zhang 的学号"+zhang.number+",zhang 求和结果"+zhang.
      f(100));
      System.out.println("henlu 的学号"+henlu.number+",henlu 求和结果"+henlu.
      f(100));
   }
}
```

p. 147 代码 4

```
interface Computable {
    final int MAX = 100;
    void speak(String s);
    int f(int x);
    float g(float x,float y);
}
abstract class A implements Computable {
   public int f(int x) {
       int sum = 0;
       for(int i=1;i<=x;i++) {
         sum = sum+i;
       }
      return sum;
    }
}
```

3. 接口的细节说明

（1）在实现接口的类中，可以通过“常量名”或“接口名.常量名”的形式使用接口的常量。

在其他类中，可以通过“接口名.常量名”的形式使用接口的常量。

（2）共有接口可以被任何类实现，友好接口只可以被在同一个包中的任何类实现。

（3）被父类实现的接口自动被子类实现。

（4）接口可以继承，从通用性较高的接口扩展到专用性较高的接口。

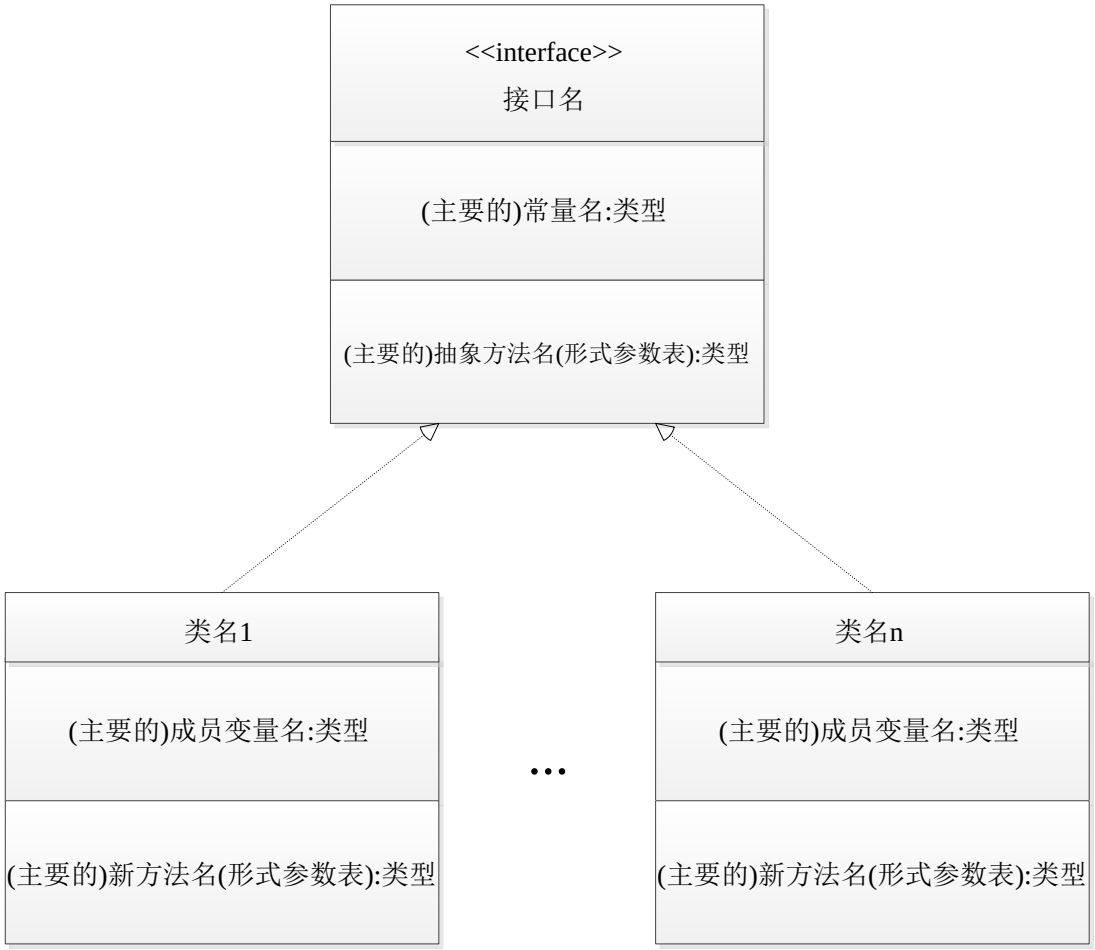
```
public interface Moveable {
    void move(double x, double y);
}
public interface Powered extends Moveable {
    double milesPerGallon();
}
```

4. 一个源文件可以包含若干类和若干接口。

6.3 接口和实现关系的 UML 图

1. 接口的 UML 图

2. 实现关系的 UML 图



6.4 接口回调

1. 声明接口变量

接口名 接口变量名 1，接口变量名 2，…，接口变量名 n;

接口变量是引用类型变量，用于存储实现接口的类的对象的引用。

2. 接口回调

（1）若接口变量存储了实现接口的类的对象的引用，则可以通过接口变量所指对象调用被类重写的接口的方法。

（2）接口回调和使用上转型对象相似。

p.150 代码 1

```
interface  ShowMessage {
   void 显示商标(String s);
}
class TV implements ShowMessage {
   public void 显示商标(String s) {
      System.out.println(s);
   }
}
class PC implements ShowMessage {
   public void 显示商标(String s) {
     System.out.println(s);
   }
}
public class Example6_2 {
   public static void main(String args[]) {
      ShowMessage sm;                    //声明接口变量
      sm = new TV();                     //接口变量中存放对象的引用
      sm.显示商标("长城牌电视机");        //接口回调
      sm = new PC();                     //接口变量中存放对象的引用
      sm.显示商标("联想奔月5008PC机"); //接口回调
   }
}
```

6.5 理解接口

实现某个接口的所有类的父类不必相同。

pp.151-152 代码 1

```
abstract class MotorVehicles {
   abstract void brake();
}
```

```
interface MoneyFare {
   void charge();
}
interface ControlTemperature {
   void controlAirTemperature();
}
class Bus extends MotorVehicles implements MoneyFare {
    void brake() {
       System.out.println("公共汽车使用毂式刹车技术");
    }
    public void charge() {
       System.out.println("公共汽车:一元/张,不计算公里数");
    }
}
```

```
class Taxi extends MotorVehicles implements MoneyFare,
ControlTemperature {
    void brake() {
        System.out.println("出租车使用盘式刹车技术");
    }
    public  void charge() {
        System.out.println("出租车:2 元/公里,起价 3 公里");
    }
    public void  controlAirTemperature() {
        System.out.println("出租车安装了 Hair 空调");
    }
}
class Cinema implements MoneyFare,ControlTemperature {
    public  void charge() {
        System.out.println("电影院:门票,十元/张");
    }
    public void controlAirTemperature() {
       System.out.println("电影院安装了中央空调");
    }
}
```

```
public class Example6_3 {
  public static void main(String args[]) {
     Bus bus101 = new Bus();
     Taxi buleTaxi = new Taxi();
     Cinema redStarCinema = new Cinema();
     MoneyFare  fare;
     ControlTemperature temperature;
     fare = bus101;
     bus101.brake();
     fare.charge();
     fare = buleTaxi;
     temperature = buleTaxi;
     buleTaxi.brake();
     fare.charge();
     temperature.controlAirTemperature();
     fare = redStarCinema;
     temperature = redStarCinema;
     fare.charge();
     temperature.controlAirTemperature();
  }
}
```

6.6 接口与多态

当接口的一个方法被多个实现接口的类重写后，可以借助接口回调，让接口的这个方法产生不同的行为。这也称为多态。

p.153 代码 1

```
interface CompurerAverage {
   public double average(double a,double b);
}
class A implements CompurerAverage {
   public double average(double a,double b) {
      double aver = 0;
      aver = (a+b)/2;
      return aver;
   }
}
class B implements CompurerAverage {
   public double average(double a,double b) {
      double aver = 0;
      aver = Math.sqrt(a*b);
      return aver;
   }
}
```

```
public class Example6_4 {
   public static void main(String args[]) {
      CompurerAverage computer;
      double a = 11.23,b = 22.78;
      computer = new A();
      double result = computer.average(a,b);
      System.out.printf("%5.2f和%5.2f的算术平均值:%5.2f\n",a,b,result);
      computer = new B();
      result = computer.average(a,b);
      System.out.printf("%5.2f和%5.2f的几何平均值:%5.2f",a,b,result);
   }
}
```

6.7 接口参数

若方法的某个形式参数是接口变量，则调用方法时相应的实际参数是实现接口的类的对象的引用。

p.154 代码 1

```
interface SpeakHello {
    void speakHello();
}
class Chinese implements SpeakHello {
   public void speakHello() {
      System.out.println("中国人习惯问候语：你好,吃饭了吗？ ");
   }
}
class English implements SpeakHello {
    public void speakHello() {
      System.out.println("英国人习惯问候语:你好,天气不错 ");
    }
}
class KindHello {
   public void lookHello(SpeakHello hello) {     //接口类型参数
      hello.speakHello();                        //接口回调
   }
}
public class Example6_5 {
   public static void main(String args[]) {
      KindHello kindHello=new KindHello();
      kindHello.lookHello(new Chinese());
      kindHello.lookHello(new English());
   }
}
```

6.8 抽象类与接口的比较

1. 若今后还打算继承一些变量和非抽象方法，则声明抽象类较好，否则声明接口较好。

2. 一个子类只可以继承一个父类，而一个类可以实现多个接口。

6.9 面向接口编程

借助接口回调，体现“开-闭原则”。

p.156 代码 1

```
public interface Advertisement { //接口
     public void showAdvertisement();
     public String getCorpName();
}
```

pp.156-157 代码 3

```
public class WhiteCloudCorp implements Advertisement { //PhilipsCorp 实现
```

```
                                                          //Advertisement 接口
    public void showAdvertisement(){
        System.out.println("@@@@@@@@@@@@@@@@@@@@@@@@");
        System.out.printf("飞机中的战斗机，哎 yes!\n");
        System.out.println("@@@@@@@@@@@@@@@@@@@@@@@@");
    }
    public String getCorpName() {
        return "白云有限公司" ;
    }
}
```

p. 157 代码 2

```
public class BlackLandCorp implements Advertisement {
    public void showAdvertisement(){
        System.out.println("**************");
        System.out.printf("劳动是爹\n 土地是妈\n");
        System.out.println("**************");
    }
    public String getCorpName() {
        return "黑土集团" ;
    }
}
```

p. 156 代码 2

```
public class AdvertisementBoard { //负责创建广告牌
    public void show(Advertisement adver) {
        System.out.println(adver.getCorpName()+"的广告词如下:");
        adver.showAdvertisement(); //接口回调
    }
}
```

p. 157 代码 3

```
public class Example6_6 {
    public static void main(String args[]) {
        AdvertisementBoard board = new AdvertisementBoard();
        board.show(new BlackLandCorp());
        board.show(new WhiteCloudCorp());
    }
}
```