第四章 | OOD

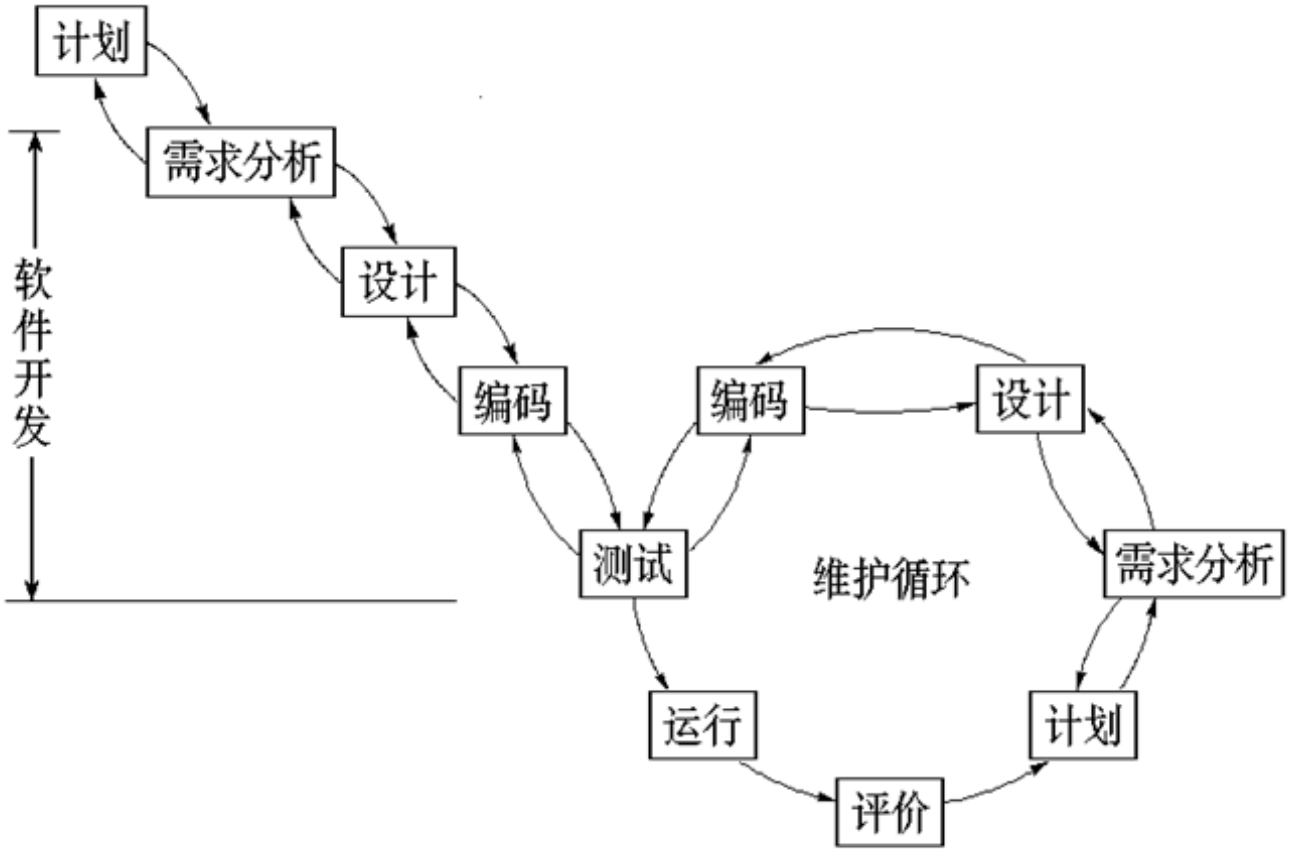
抽象类和接口

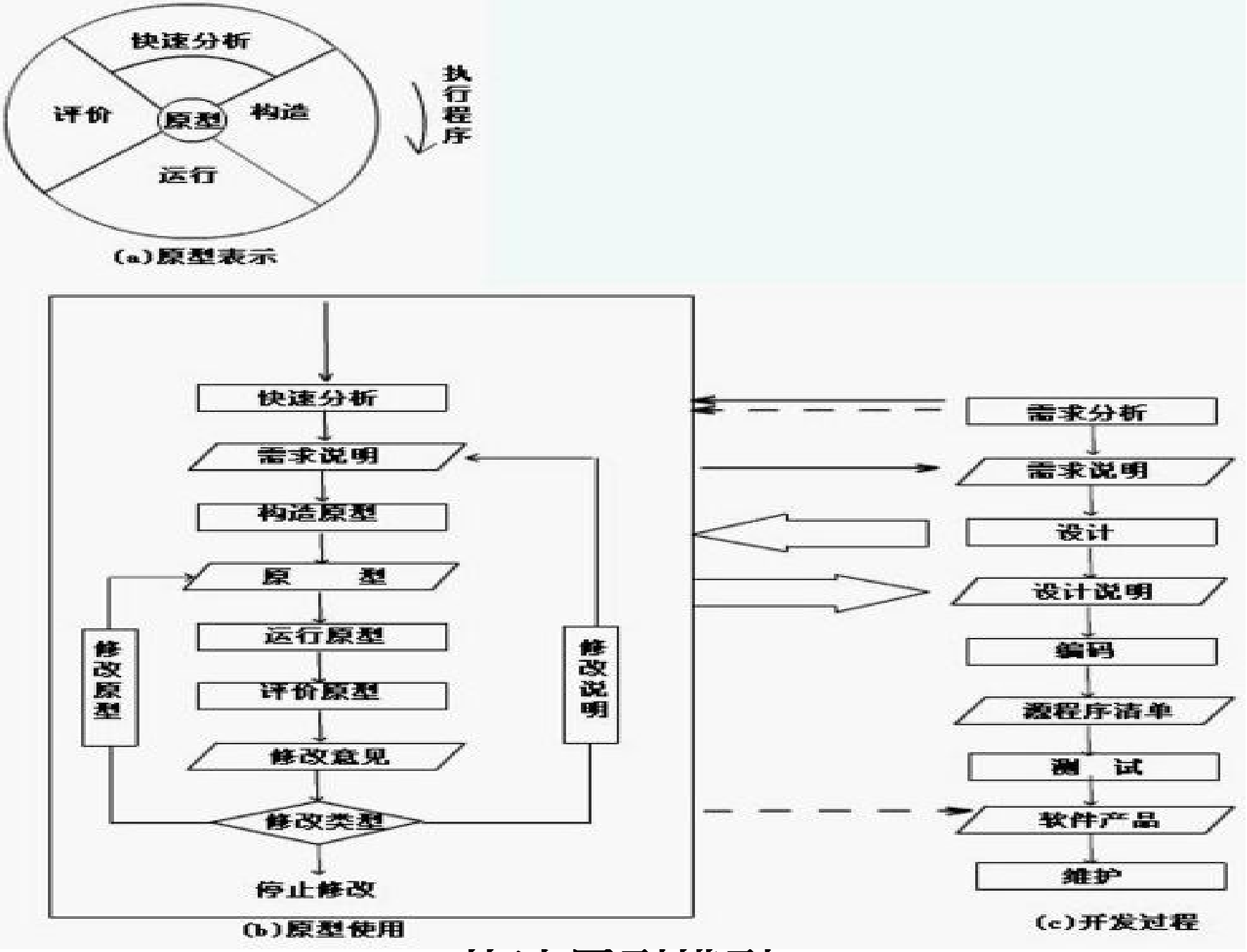
涉及到课本的章节:

- 第3章 类的封装、继承和多态
- 第4章 接口

OOD

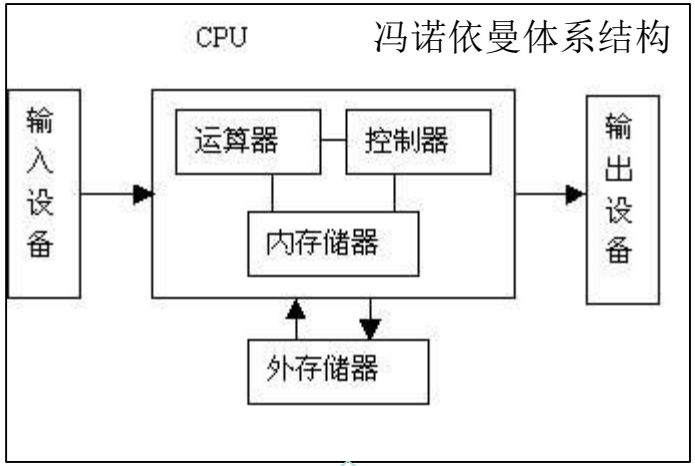
- 软件产品的产生过程（瀑布模型）




快速分析
评价
原型
构造
运行
执行程序
(a)原型表示
快速分析
需求说明
构造原型
原　　型
运行原型
评价原型
修改意见
修改类型
停止修改
修改原型
修改说明
(b)原型使用
需求分析
需求说明
设计
设计说明
编码
源程序清单
测　试
软件产品
维护
(c)开发过程

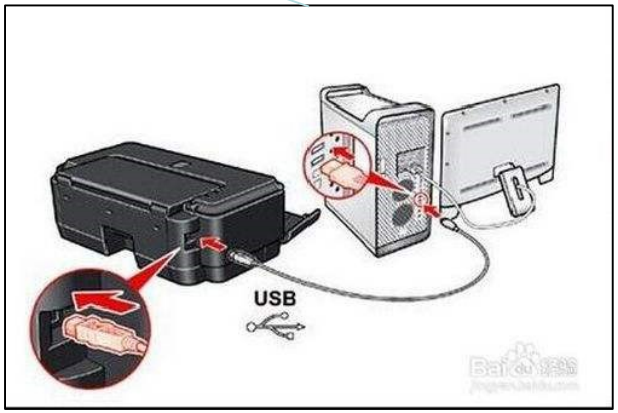
快速原型模型

抽象类和接口

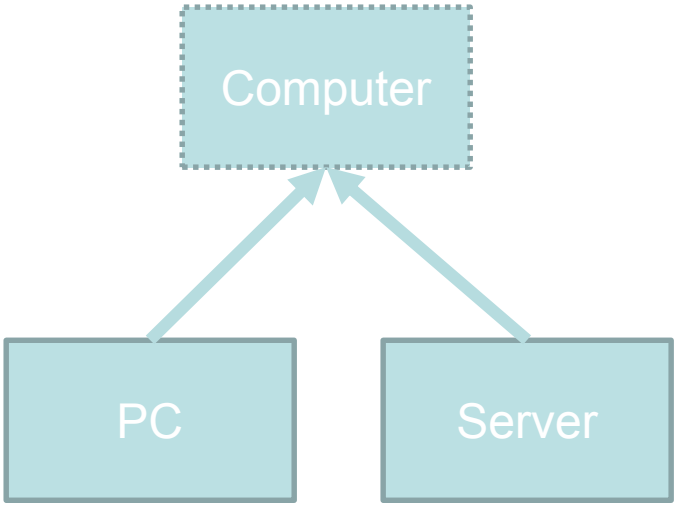

CPU
冯诺依曼体系结构
输入设备
运算器
控制器
内存储器
外存储器
输出设备

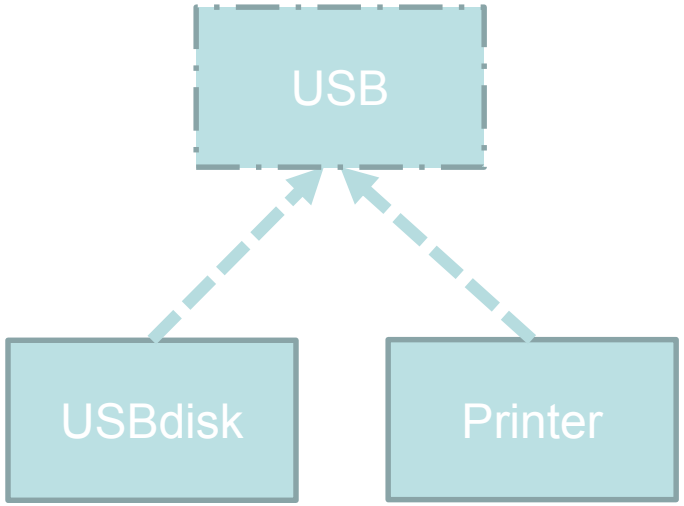

SUPERSPEED+
CERTIFIED
USB
10Gbps
USB 3.1


GND
D+
D-
VCC
StdA_SSRX-
StdA_SSRX+
StdA_SSTX+
StdA_SSTX-
GND_DRAIN
USB 2.0
USB 3.0

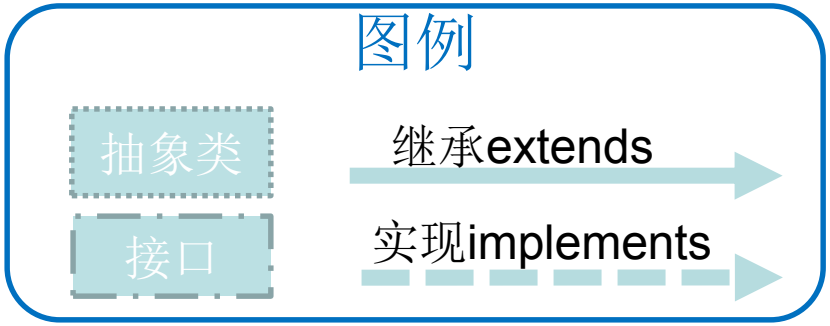

USB

抽象类与接口





接口实现的内容并非核心功能
一般也不满足类继承的is a验证



为什么将二者放在一起介绍?

数据类型:

二者都是引用数据类型.

语言规范:

(1)二者都是用来完成抽象定义的.接口是一种特殊的抽象类

(2)二者都是用来定义一种规范

(3)二者都涉及到OOD(Design)

第四章I 抽象类和接口

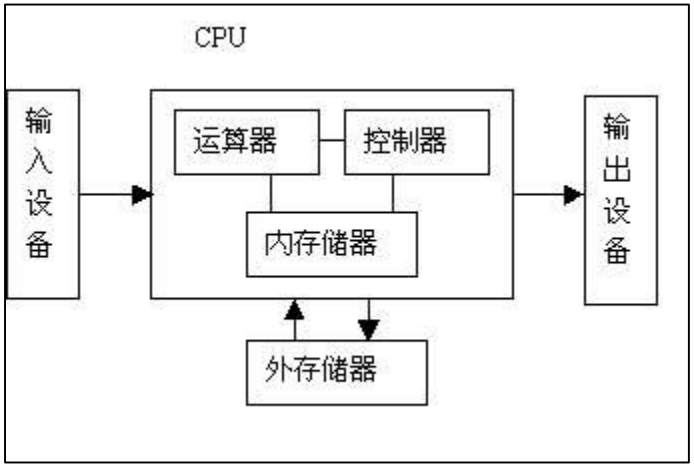
- 1. 抽象类
  - (1) 抽象类定义
- 2. 接口
  - (1) 接口定义
  - (2) 实现接口
  - (3) 接口继承
- 3. 现实应用举例

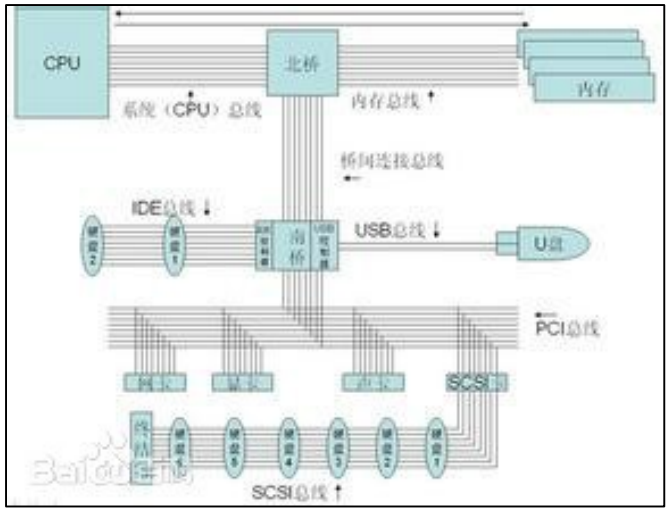
1、抽象类概念

- 在面向对象的概念中，万事万物皆对象，所有的对象都是通过类来描绘的。
- 但是反过来并不成立，并不是所有的类都是用来描绘对象的，如果一个类中没有包含足够的信息来描绘一个具体的对象，需要一个子类继承它之后才能描绘一个对象，这样的类就是抽象类。
- 【总结】通常Java中类是描绘对象的，但是有这么一些特殊的类，不能描绘对象但是能专门描绘子类

抽象类引例解析

- 计算机抽象类不能描绘对象，虽然定义了框架（如CPU含运算器+控制器+内部存储器）但并不能据此打造计算机，打造计算机还需其他内容，如总线、输入输出设备接口等





1. 抽象类定义

- 抽象类:

  ---本身不能被实例化对象,只能作为其它类的超类

  ---使用abstract修饰

- 定义:

```
abstract class 类名{
  属性定义;
  方法定义;      // [抽象方法]  非抽象方法
}
```

引出:抽象方法定义

```
abstract 返回类型 方法名();
```

抽象类示例:

```
abstract class Animal{
    private String name;                    //成员变量定义
    private int age;
    public Animal(){   }
    public Animal(String name,int age){
      this.name=name;
      this.age=age;
    }                                       //构造方法定义
    public void makeSound() ;               //抽象方法定义
    public void toString()                  //实例方法定义
       {System.out.println(name+"now age is"+age);}
}
```

抽象类定义规范:

- 1含有抽象方法的类必须被定义成抽象类
- 2抽象类中不一定要有抽象方法
- 3构造方法不能用abstract

- 抽象类用处:

--被子类继承

- 抽象类继承规范:

1抽象类被子类继承,必须具体实现抽象方法

2子类必须实现抽象父类的所有抽象方法

抽象类举例1：

- 问题描述：

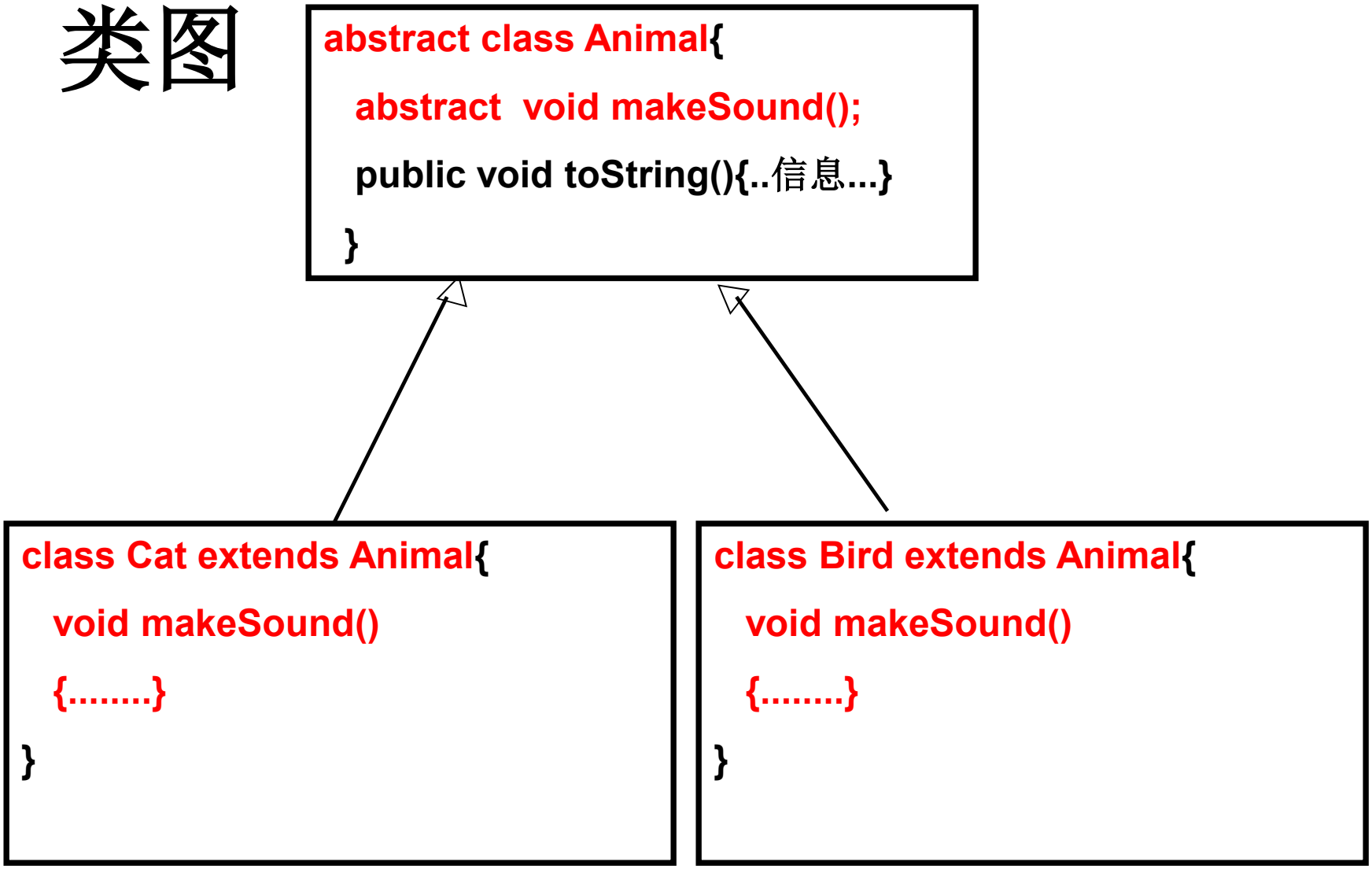
现需要设计一个动物类，对不同的动物(猫,狗,兔子,鸟等)的叫声能提供一个统一的规范

问题解决:

将**Animal**类定义成抽象类;

将叫声定义成一个抽象方法。

类图


abstract class Animal{
abstract void makeSound();
public void toString(){..信息...}
}
class Cat extends Animal{
void makeSound()
{.....}
}
class Bird extends Animal{
void makeSound()
{.....}
}

抽象类举例2：

- 问题描述：

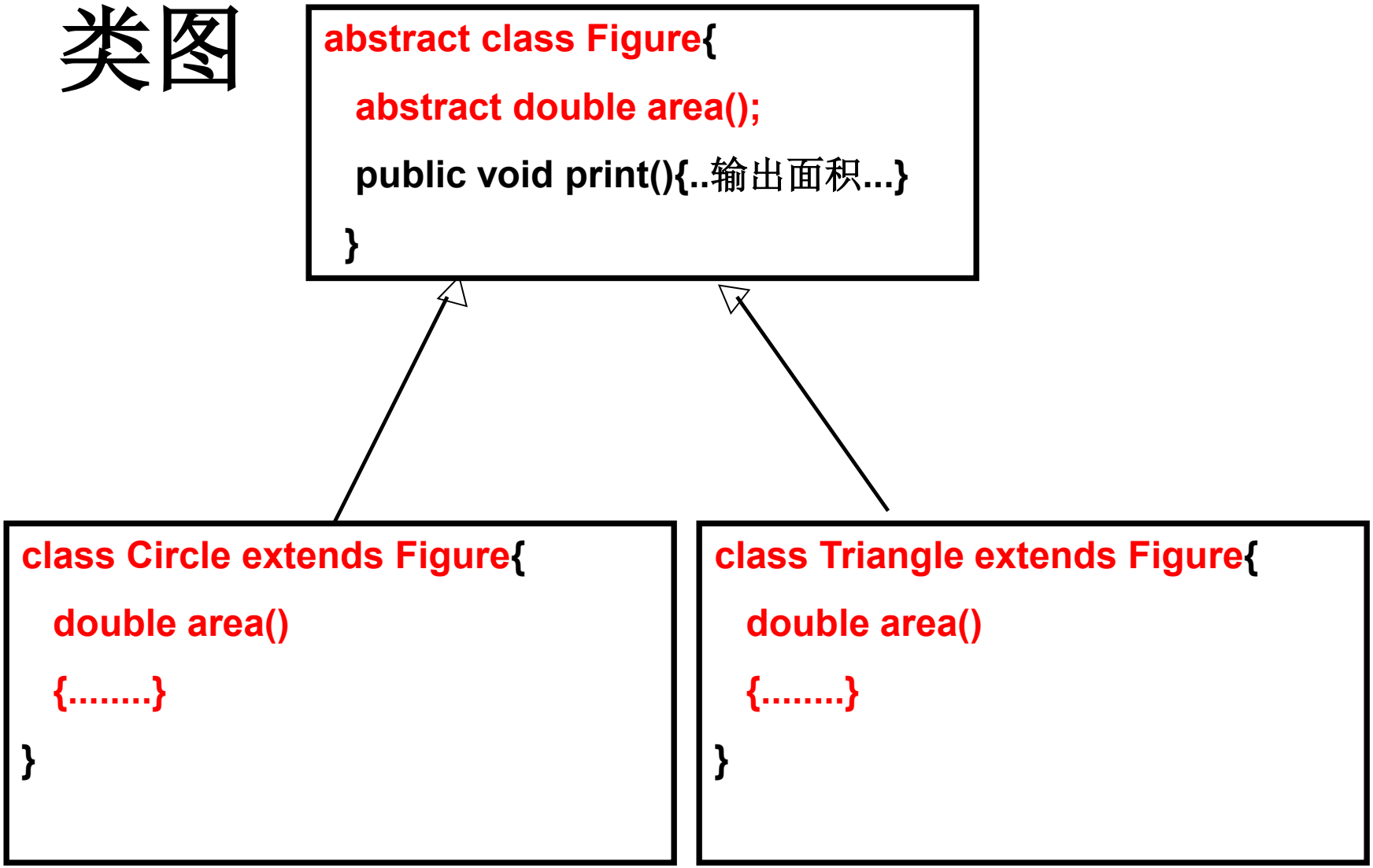
现需要设计一个图形类Figure，对不同的图形(三角形，圆形等)求面积能提供一个统一的规范

问题解决:

将Figure类定义成抽象类;

将求面积定义成一个抽象方法。

类图


abstract class Figure{
abstract double area();
public void print(){..输出面积...}
}
class Circle extends Figure{
double area()
{.....}
}
class Triangle extends Figure{
double area()
{.....}
}

抽象类举例:

```
abstract class Figure
  {  private String shape;
     public Figure(String shape)
          { this.shape=shape;}
     public  abstract double area();
     public void print()
     {   System.out.println(this.shape+”面积为:”+this.area());
     }
  }
```

```
class Circle extends Figure
•   {  double r;
•      Circle(double r)
•        {  super(“圆形”);
•           this.r=r;
•        }
•      public double area()
•       {   return(3.14*r*r);
•        }
•   }
```

```
class Triangle extends Figure
  {
 double sideA,sideB,sideC;
 boolean boo;
 Triangle(double a,double b,double c)
 {  super(“三角形” );
    sideA=a;
    sideB=b;
    sideC=c;
   if(a+b>c&&a+c>b&&c+b>a)   {
     System.out.println("我是一个三角形");
     boo=true;
     }
   else
   {
     System.out.println("我不是一个三角形");
     boo=false;
     }
    return bool;
 }
```

```
public double area()
 {
    if(boo)
       {
          double p=(sideA+sideB+sideC)/2.0;
          double area=Math.sqrt(p*(p-sideA)*(p-
sideB)*(p-sideC)) ;
          return area;
       }
      else
       {
         System.out.println(“不是一个三角形,不
能计 算面积");
         return 0;
       }
     return 0;
   }

}//end of class Triangle
```

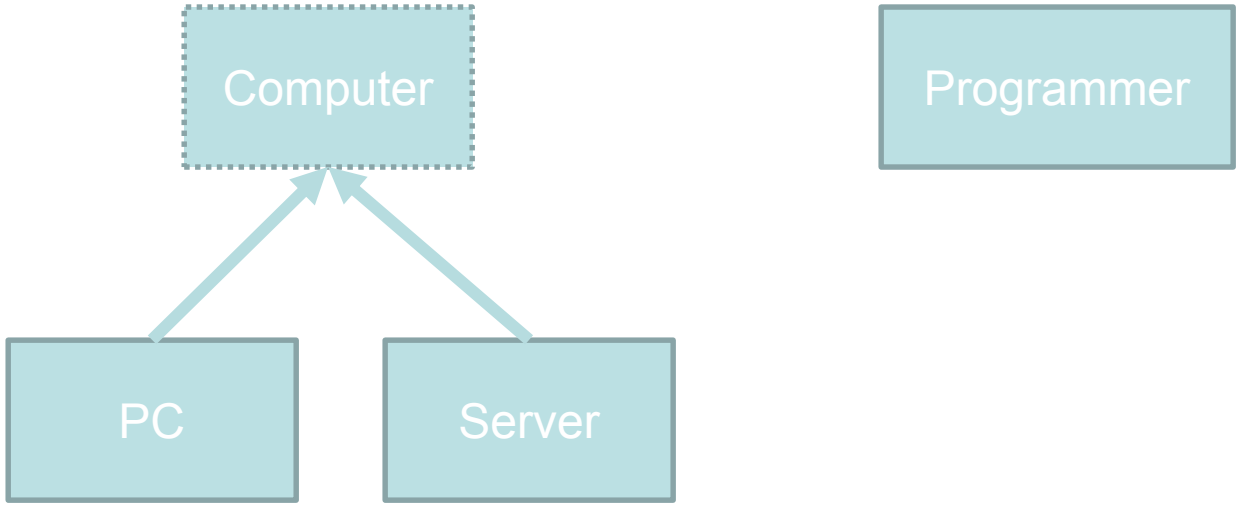
```
class AbstractDemo
 {
     public static void main(String[ ] args)
   {
        Figure mycircle=new Circle(2);
        mycircle.print();
        Figure mytriangle=new Triangle(3,4,5);
        mytriangle.print();
   }
 }
```

抽象类改造升级

- 需要求图形的周长

```
abstract class Figure{
   abstract double area();
   public void print(){..输出面积...}
   abstract double perimeter();
  }
```

抽象类示例（Computer）



cn.sdu.java.abstractClass.computer

抽象类意义:

- 抽象类的抽象方法约定了多个子类共用的方法声明.
- 不同的子类可以有不同的抽象方法实现,这样体现了OO的多态性.
- 可以使子类都在同一个规范约束下.