**抽象类的定义与使用**

1. 抽象类基本概念
   1. 类继承的主要功能是用于扩充已有类的功能，但是对于之前学习的类继承，子类可以自有选择覆写方法，父类无法强制性要求子类覆写某个方法。在实际开发中我们一般很少继承已经完善的类(class Person() )，可以直接使用的类，而是必须继承抽象类（abstract类），在类的前面增加关键字 abstract 便是抽象类，抽象类无法直接使用new 实例化对象。在以后的父类设计中一定优先考虑的是抽象类。
2. 抽象类的基本定义
   1. 抽象类的主要作用是在子类中对覆写方法做约定，在抽象类中可以定义一些抽象方法去实现这样的约定，使用了 abstract 的方法并且没有提供方法体的便是抽象方法，而抽象方法所在的的类就是抽象类。(在普通方法中追加抽象方法) 也就是说抽象类 和其它的普通类除了能够定义抽象方法 其它的都是一样的。

|  |
| --- |
| abstract class Person{  public abstract String getName();  }  class son extends Person{  public String getName(){  return "我是李四";  }  }  public class JavaDemo{  public static void main(String [] arg){  Person per = new son();  //Person per = new Person(); 错误  System.out.println(per.getName());  }  } |

注意在该案例中有一个错误示例： 抽象类无法直接new 实例化对象。

* 1. 在一个抽象类定义完成后需要注意以下几点
     1. 抽象类不是一个完整的类，如果想要去使用抽象类，我们必须要遵循以下几点
        1. 抽象类必须有子类，子类使用extends 继承
        2. 抽象类的子类（不是抽象类）必须覆写抽象类的全部方法。
        3. 抽象类的实例化可以通过对象的多态性通过子类向上转型的方式完成
  2. 对于抽象类使用的几点意见。
     1. 抽象类有一个很大的核心问题，无法自己new 实例化对象。
     2. 抽象类目的主要是用于过渡操作使用，所以当我们要使用抽象类进行开发的时候，往往都是在你设计中需要解决类继承问题时所带来的代码重复操作。

1. 抽象类的相关说明
   1. 抽象类在使用时有以下几个注意点
      1. 在定义抽象类的时候我们绝对不能够使用final 关键字来进行定义，因为抽象类必须要用子类，而final 类不能有子类，因此不能使用final 去定义抽象类。
      2. 抽象类作为一个普通类的加强版（抽象类就是在普通类上进行扩展，追加了抽象方法）既然是在普通类基础上进行扩展，普通类中就可以定义属性方法，那么这些属性一定是要求开辟空间的，所以抽象类可以提供构造方法，而且子类一定会按照实例化原则进行父类构造调用。

|  |
| --- |
| abstract class Person{  private String type;  public abstract String getName();  public Message (String typr) {  this.type = type ;  }  }  class son extends Person{  public son (String type){  super(type);  }  public String getName(){  return "我是李四";  }  }  public class JavaDemo2{  public static void main(String [] arg){  Person per = new son();  System.out.println(per.getName());  }  } |

* + 1. 抽象类中即便没有抽象方法，也不能直接使用new 实例化抽象类，必须依靠子类对象完成
    2. 抽象类中可以提供有static 方法，并且不受抽象类影响；

|  |
| --- |
| abstract class Person{  private String type;  public abstract String getName();  public static Person getInfo(){  return new son();  }    }  class son extends Person{  public String getName(){  return "我是李四";  }  }  public class JavaDemo2{  public static void main(String [] arg){  Person per = Person.getInfo();  System.out.println(per.getName());  }  } |

Static 方法不受对象实例化的影响，永远可以通过类名称进行调用。

1. 抽象类的应用(模板设计模式)
   1. 首先我们应该明确抽象类是比普通类更加高级的一层；
      1. 抽象类是一是:用对与子类进行统一管理的类，二是: 自身提供有一些普通方法，并且这些方法可以调用抽象方法。(这些抽象方法必须在子类提供实现的时候才会生效)

|  |
| --- |
| abstract class Action{  public static final int EAT = 1;  public static final int SLEEP = 5;  public static final int WORK = 10;  public void commd(int commd){  switch (commd){  case EAT:{  this.eat();  break;  }  case SLEEP:{  this.sleep();  break;  }  case WORK:{  this.work();  break;  }  }  }  public abstract void eat();  public abstract void work();  public abstract void sleep();    }  class Robot extends Action{  public void eat(){  System.out.println("机器人需要电");  }  public void sleep(){  }  public void work(){  System.out.println("机器人进行无思想的工作");  }  }  class Person extends Action{  public void eat(){  System.out.println("人需要食物");  }  public void sleep(){  System.out.println("人需要睡觉");  }  public void work(){  System.out.println("人进行有思想的工作");  }  }  class pig extends Action{  public void eat(){  System.out.println("猪吃剩饭");  }  public void sleep(){  System.out.println("猪倒地就睡");  }  public void work(){  }  }  public class JavaDemo3{  public static void main(String [] arg){  Action robotAction = new Robot();  Action personAction = new Person();  Action pigAction = new pig();  robotAction.commd(1);  }  } |

案例：javaDemo3