2022年6月12日　　9:16

抽象类

2022年6月6日　　9:32

理解abstract类

1. 抽象类可以抽象出重要的行为准则，该行为准则用抽象方法表示，即抽象类封装了子类必须要有的行为准则
2. 抽象类声明的对象可以成为其子类的对象的上转型对象，调用子类重写的方法，体现子类根据抽象类中的行为准则给出具体行为

面向抽象编程

使用多态进行程序设计的核心技术之一是使用上转型对象；
软件设计面临的最大问题是用户需求的变化

面向抽象编程的核心：

让类中每种可能的变化对应地交给抽象类的一个子类去负责

开-闭原则

将应对用户变化的部分设计为对扩展开放的，而设计的核心部分是对修改关闭的

抽象类与抽象方法

2022年6月6日　　9:33

abstract类和abstract方法

abstact类（抽象类）

```
abstract class A{
    ……
}
```

abstact方法（抽象方法）

```
abstact int min(int x,int y)
```

对于抽象方法，只允许声明，而不允许实现（没有方法体），不允许使用final和abstract同时修饰一个方法或类，且abstrct方法必须是非private类的实例方法

抽象的特点

2022年6月6日　　9:35

抽象的特点

1. abstract类中可以有abstract方法：非抽象类中不可以有抽象方法，抽象类中既可以有非抽象方法，也可以有抽象方法

2. abstract类不能用new标识符创建该类的对象

3. 如果某个非抽象类是某个抽象类的子类，那么它必须重写父类的抽象方法（这是不允许final修饰抽象方法的原因）

4. 抽象方法因为没有方法体，所以它主要的作用就是被重写(不允许final修饰抽象类）

5. 一个抽象类是另一个抽象类的子类，那么该抽象类既可以重写也可以继承另一个抽象类

6. 可以使用abstract类声明声明对象，该对象可以成为其子类对象的上转型对象，那么该对象就可以调用子类重写的方法

利用上转型对象进行传值

2022年6月6日　　9:40

利用上转型对象进行传值

```
class boy {
        GirlFriend friend;
        void setGirlfriend(GirlFriend f) {
            friend = f;
        }
}
public static void main(String[] args) {
              GirlFriend girl = new ChinaGirlFriend();    //抽象类创建的对象是上转型对象
              boy boy = new boy();
              boy.setGirlfriend(girl);
}
```

接口

2022年6月6日　　9:43

接口定义　　接口声明+接口体：

```
interface Com{
        ……
}
```

类实现接口

```
class A implements Com , Addable
class Dog extends Animal implements Eatable , Sleepable"
```

函数接口　　如果一个接口只有一个abstract方法，称这样的接口为单接口，也称为函数接口

Lambda表达式

普通表达式：

```
int computeSum(int a,int b){
        return a + b;
    }
```

Lambda表达式：

```
(a,b)->{
    return a + b;
    }
```

即：(参数列表)->{

　　方法体

}

Lambda表达式的值　就是方法的入口地址

接口变量存放Lambda表达式的值

对于函数接口，允许把Lambda表达式的值赋值给接口变量，那么接口变量就可以调用Lambda表达式实现的方法

也就是说，Lambda表达式实现了该函数接口中的抽象方法，并将所实现的方法的入口地址化为此Lambda表达式的值

理解接口

1.接口可以抽象出重要的行为准则，该行为准则用抽象的方法表示

2.可以把实现接口的类的对象的引用赋值给接口变量，该接口变量可以调用被该类实现的接口方法

接口在要求一些类有相同名称的方法的同时并不强迫这些类具有相同的父类

允许在接口体中定义 static 方法。例如，下列接口中的 f() 方法就是

```
int MAX = 100;
 on();
t sum(float x,float y);
ax(int a,int b) {

() {  //static 方法
ln("注意是从 Java SE 8 开始的");
```

e 方法

允许在接口体中定义 private 的方法，其目的是配合接口中的 default 实例方法调用某些算法封装在 private 的方法中，供接口中的 default 实例方法调用。

类来实现，以便使用接口中的方法。一个类需要在类声明中使用关

实现一个或多个接口，如果实现多个接口，用逗号隔开接口名，例

- 需在接口中增加新的方法，但并非所有的原实现类都需该方法，则用 default.
- 欲将为接口中都有b()方法，引用父类中的b()方法.
- 所继承的两个接口都有b()方法，则需重写b()方法.
- Math类的构造方法是private的
- 接口中的static方法需接口名访问
  static变量可用3种方法访问.

接口体

2022年6月6日 9:50

接口体

1. 可以有抽象方法和常量；抽象方法的访问权限一定是public，而且可以省略abstract和public修饰符

2. static常量的访问权限一定是public，而且可以省略public、final、static修饰符

3. 接口体中不会有变量

4. 可以定义default实例方法，default的实例方法和普通的实例方法相比就是用关键字default修饰的带方法体的实例方法

5. 不能省略关键字default，因为不允许定义普通的带方法体的public实例方法

6. 可以定义static方法

7. 可以定义private方法，其目的是配合default实例方法，即接口可以将某些算法封装在private的方法中，供接口中的default实例方法调用，实现算法的复用

重写接口中的方法

2022年6月6日　　9:52

重写接口中的方法

1. 如果一个类实现了某个接口，那么这个类就自然拥有了接口中的常量、default方法（去掉了default关键字）

2. 该类可以重写接口中的default方法（重写时去掉default关键字）

3. 如果一个非abstract类实现了某个接口，那么这个类就必须重写该接口的abstract方法，称为类实现的接口方法

4. 如果一个abstract类实现了某个接口，那么该类可以选择重写接口的abstract方法或直接拥有接口的abstract方法

使用接口中的常量和static方法

2022年6月6日　　9:54

可以用接口名访问接口中的常量、调用接口中的static方法：Com.MAX;
Com.f();

常量可以用接口名或所实现接口的类的类名或类所创建的对象调用

而static方法只能由接口名调用

接口中的细节

2022年6月6日　　9:55

接口的细节

如果一个接口不加public修饰，就称友好接口，友好接口可以被与该接口在同一个包中的类实现

如果父类实现了某个接口，那么子类也自然实现了该接口，子类不必再显式地实现该接口

接口也可以被继承，即可以通过关键字extends声明一个接口是另一个接口的子接口：由于接口 中方法和常量都是public的，子接口将继承父接口中全部实例方法和常量

import java.io.*　不仅引入了包中的类，同时也引入了该包中的接口

接口回调

2022年6月6日　　9:58

接口回调

用接口声明的变量称为接口变量

在接口变量中可以存放实现该接口的类的实例的引用，即存放对象的引用

当把对象的引用赋值给接口变量后，那么该接口变量就可以调用被类实现的接口方法以及接口提供的default方法或类重写的default方法；

实际上，当接口变量调用被类实现的接口方法时就是通知相应的对象调用这个方法

接口回调非常类似于上转型对象调用子类重写的方法；

接口无法调用类中的其他非接口方法

接口参数

如果一个方法的参数（形参）是接口类型，那么就可以将任何实现该接口的类的实例的引用传递给该接口参数，
那么该接口参数就可以回调类实现的接口方法
如果参数是函数接口，也可以将Lambda表达式的值传递给该接口参数

abstract类与接口的比较

2022年6月6日　　10:29

如果子类除了要重写父类的abstract方法以外，还需要从父类继承一些变量或继承一些重要的非abstract方法，就可以考虑用abstract类；如果某个问题不需要继承，只需要若干个类给出abstract的某些细节，就可以考虑接口