Java基础

*反射：*

类是对象，类是java.lang.Class类的实例对象，任何一个类都是class的实例对象，有三种表示方式

方式1：类.class

方式2： 类实例.getclass()

方式3：Class.forname(“类的全路径”)

1. Class类被final修饰，不能被继承
2. Class的构造方法被private修饰，不能通过new来得到实例对象

A a = new A();

Class c1 = a.getClass();

Class c2 = A.Class;

a代表A的实例对象， c1代表class类的实例对象，我们称

为该类的类类型（c1表示了A的类类型（class type））

C1==c2---> 因为一个类只有一个类类型

类类型得到类的实例对象： A a =（A）c1.getInatance（）---->需要有无参构造方法，或则先得到有参构造函数，然后再通过getInatance方法得到类的实例对象

Class<ReflectDemo> a1 = ReflectDemo.class;  
Constructor<ReflectDemo> constructor = a1.getConstructor(int.class);  
ReflectDemo a2 = constructor.newInstance(5);

*Class类动态加载*

编译时刻加载类是静态加载类，运行时刻加载类是动态加载类

New属于静态加载

Class.forname属于动态加载类

反射的操作都是编译之后的操作，属于字节码的操作

Arraylist list = new Arraylist（）；

Arraylist list1 = new Arraylist（）；

List == list1--->为true

说明编译之后的集合去除了泛型，编译之后集合无泛型

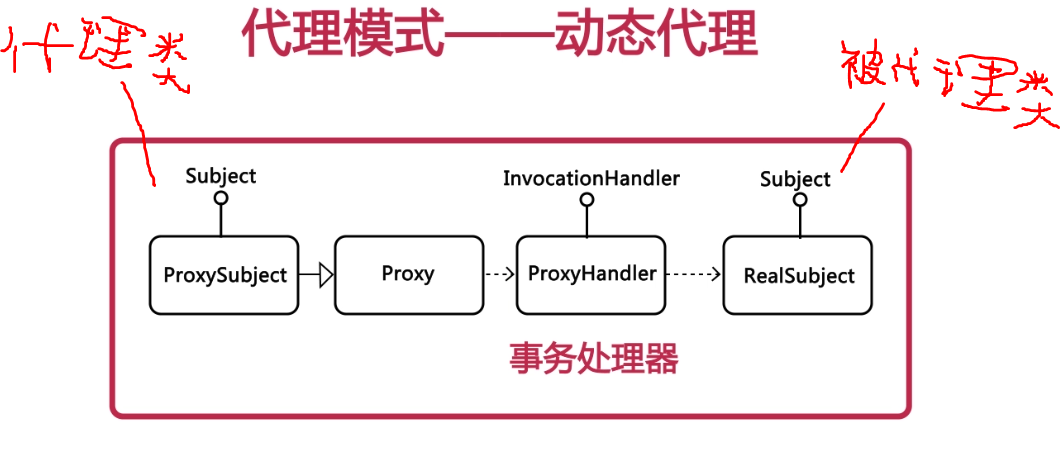
Java中集合的泛型是防止错误输入的，只在编译期有效

代理模式

*静态代理*

1. 继承
2. 聚合---->推荐使用

*JDK动态代理（只能代理实现了接口的类）*

**

*Invocationhandler：*

***Invoke(object obj,method meth,Object[] args)***

***Obj指示代理对象***

***Meth被代理的方法***

***Args为被代理方法的参数***

*Proxy：*

***Proxy.newProxyInstance(loader,interface,h)***

***Loader:被代理的加载器***

***Interface：被代理的类实现了哪些接口***

***H为前面的Invocationhandler***

实现步骤：

public class TimeLogImpl implements InvocationHandler {

private Object target;

public TimeLogImpl(Object target){

this.target = target;

}

@Override

public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {

System.out.println("开始");

method.invoke(target);

System.out.println("结束");

return null;

}

}

1. 实现invocationhanlder，重写invoke方法
2. 通过传入目标对象，供invoke方法中method参数使用
3. 利用invoke方法中的method方法进行对被代理类方法的操作
4. 在mehod.invoke方法前后进行逻辑增强

Proxy的实现

1. proxy.newproxyinstance得到proxy代理类
2. 提供newproxyinstance中的参数
3. 利用得到的代理类调用目标方法

public static void main(String[] args) {

ProxyAimsCar proxyAimsCar = new ProxyAimsCar();

Class<? extends ProxyAimsCar> aimClass = proxyAimsCar.getClass();

TimeLogImpl timeLog = new TimeLogImpl(proxyAimsCar);

/\*

public static Object newProxyInstance(ClassLoader loader,

Class<?>[] interfaces,

InvocationHandler h)

throws IllegalArgumentException

\*/

CarInterFaceDemo carInterFaceDemo = (CarInterFaceDemo) Proxy.newProxyInstance(aimClass.getClassLoader(), aimClass.getInterfaces(), timeLog);

carInterFaceDemo.run();

}



*Cglib动态代理（针对类来实现代理）*

jxl实现数据写入/读取excel

1. 创建文件 file f = new file（“指定路径”） f.createnewfile
2. 创建writableWorkbook----->利用得到的writableWorkbook创建writeableSheet
3. 设置表头
4. 设置内容

String [] s = new String[] {"sex","age","sc"};

WritableWorkbook writableWorkbook = Workbook.*createWorkbook*(file);  
WritableSheet sheet1 = writableWorkbook.createSheet("sheet1", 0);  
Label label = null;

设置表头  
for (int i = 0 ;i<s.length;i++){  
 //列，行，值 第i列第0行的值为s[i]  
 label = new Label(i,0,s[i]);  
 sheet1.addCell(label);  
 }

设置内容  
for (int i =1;i<30;i++){  
 label = new Label(0,i,"男");  
 sheet1.addCell(label);  
 label = new Label(1,i,"19"+i);  
 sheet1.addCell(label);  
 label = new Label(2,i,"cz");  
 sheet1.addCell(label);  
}

写入数据，关闭流

writableWorkbook.write();  
 writableWorkbook.close();

读取数据：

Workbook.getworkbook---> 得到文件

Workbook workbook = Workbook.*getWorkbook*(new File("C:/Users/YongShengZhang/Downloads/test\_demo.xls"));  
  
通过得到的文件得到sheet表格，操作表格的行列得到cell单元的数据  
Sheet sheet = workbook.getSheet("0");  
for (int i=0;i<sheet.getRows();i++){  
 for (int j = 0; j<sheet.getColumns();j++){  
 Cell cell = sheet.getCell(i, j);  
 System.*out*.println(cell);

得到单元格的值

System.*out*.println(cell.getContents());

== 比较对象，比较的堆内存的地址是否相同，一般用来比较基本类型的值是否一样，equals重写之后是比较的内容是否一样，hashcode返回的是对象hash值