试题-带答案

前言：

1. 下面的题目不包含JDBC和网络编程， 后续我们会有大作业。

2. 题目中只要不要求编程， 不要实际去运行程序。

3. 除非特别注明，每道题均是1分

4. 请自测得分， 然后把结果输入到

https://shimo.im/spreadsheet/DGZftdwmBCUpf4mg

注意，不要修改别人的得分，以免混乱

第一部分： Java 概述

1.1 Java 是怎么做到跨平台的？

要点：

1. Java 源文件编译成了自定义的平台无关的字节码

2. 为Windows，Linux, Mac等不同平台开发虚拟机来执行字节码

1.2 Java EE, Java SE, Java ME分别是什么？

Java 企业版，标准版，移动版

1.3 简述JRE与JDK的区别?

JRE: Java Runtime Environment; Java运行环境，面向用户

JDK: Java Development Kits; Java开发工具包，面向开发者，JDK中包含有JRE.

1.4 为什么要设置Java Home 和Path？

要点：

某些应用程序如果依赖JRE/JDK， 会在运行时利用Java_Home指定的路径来搜索

设定Path可以帮助在调用java.exe/javac.exe的时候不用使用全路径（例如c:\java8\bin\java.exe ）

直接用java xxxx 就可以了。

设置了java_home, 也可以方便设置path， <java_home>/bin

1.5 在运行java 程序时，为什么要设置classpath？

要点：Java 虚拟机的类装载机制会通过classpath来寻找类并加载， 如果没有设定classpath, 会出现类找不到的情况。

1.6 一个.java源文件是否可以包含多个类？有什么限制

可以， 但是只能有一个类是public 的。

第二部分： 语法

2.1 以下数据类型分别占多少字节？

byte 1
short 2
int 4
long 8
float 4
double 8

2.2 下面语句有什么错误？

```
short s1 = 1;
s1 =s1 + 1;
```

编译错误： s1+1，编译器会认为生成int 类型， 不能把int 转化为 short

2.3 int和Integer都可以代表一个整数， 为什么JDK设计了这两个类型？

要点： int是个原生类型， 不是一个类，不能和其他面向对象的类进行交互，例如加入到一个ArrayList 中。

Integer 是对int 的一个封装

2.4 在脑子里运行下面的代码， x的值最终是多少？

```
public class JavaTest {
    public static void main(String args[]) {
        int x=2 ;
        do{
            x+=x ;
        }
        while(x<17);
        System.out.println(x);
    }
}
```

答案：x = 32

2.5 请看下面的程序代码：

```
switch(n){
    case 0: Syste.out.println(“first”);
    case 1:
    case 2: Syste.out.println(“second”); break;
    default: Syste.out.println(“end”);
}
```

当n为何值时，程序段可以输出字符串”second”

A. 0　　B. 1　　C．2　　D．以上都可以

答案： D

2.6 请问 下面的代码执行完毕以后， b的值， x的值，y的值各是多少？

```
int x=6,y=8;
boolean b;
b=x>y || ++x==--y;
```

答案： b 为 true , x, y 均为7

2.7 请问 下面的代码执行完毕以后， b的值， x的值，y的值各是多少？

```
int x=6,y=8;
boolean b;
b=x>y  &&  ++x==--y;
```

答案： b 为 false, x为6， y 为8

2.8 下面的程序输出是什么？

```
public class JavaTest {

    public static void changeStr(String str){
        str="welcome";
    }
    public static void main(String[] args) {

        String str="1234";
        changeStr(str);
        System.out.println(str);
    }
}
```

答案： 1234

2.9 有代码如下， 请问调用changeValue以后， 数组arr的值是多少？

```
public class JavaTest {
 private static void changeValue(int[] arr) {
  for (int i = 0; i < arr.length; i++)
   arr[i] *= 2;
 }
 public static void main(String[] args) {
  int[] arr = { 1, 2, 3, 4, 5 };
```

```
  changeValue(arr);

 }
}
```

答案：{ 2, 4, 6, 8,10}

2.10 下面代码有什么错误？

```
public class JavaTest {
   final int i;
   public void doSomething() {
      System.out.println("i = " + i);
   }
}
```

答案： int i 为final , 但是没有初始化

2.11 下面的代码试图计算 2.00 - 1.10, 但是输出的值确是0.8999999999999999 ,这是为什么？

```
public class JavaTest {
 public static void main(String[] args) {
  System.out.println(2.00-1.10);
 }
}
```

答案： 因为浮点类型是不精确的， 无法精确的表示小数， 要精确的话需要BigDecimal

第三部分： 面向对象

3.1 下面代码有什么错误？

```
public class JavaTest {
    public static void main(String[] args) {
      JavaTest s = new JavaTest();
      System.out.println("s.doJavaTest() returns " + doJavaTest());
   }
    public String doJavaTest() {
      return "Do JavaTest ...";
   }
}
```

答案： 静态方法不能引用非静态方法doJavaTest();

3.2 下面的代码有什么问题？ 为什么？

```
public class JavaTest {
 void doSomething() {
  private String s = "";
  int l = s.length();
 }
}
```

答案： private 不能用于局部变量

3.3 试完成下述程序片段

```
public class Point{
 int x,y;
 public  Point(int x,int y){
  (  )=x;
  (  )=y;
   }
   ...
```

}
A . Point.x Point.y B . this.x this.y
C . super.x super.y D . 无解

答案： B

3.4 下面的代码有什么问题？

```
public class JavaTest {
 public static void main(String[] args) {
  Hello obj = new Hello();
  obj.msg += ",World!";
  System.out.println(obj.msg);
 }
}

class Hello {
 public String msg = "Hello";
 public Hello(String msg) {
  this.msg = msg;
 }
}
```

答案： 类Hello 没有缺省构造函数, 无法使用 new Hello();

3.5 下面的代码有什么问题？ 为什么？

```
public class JavaTest{
  public int addOne(final int x) {
    return ++x;
  }
}
```

答案： x 是final 类型， 不能做加1操作

3.6 下面的代码有什么问题？ 为什么？

```
public class JavaTest{
  public static void main(String[] args) {
    Other o = new Other();
    new JavaTest().addOne(o);
  }
  public void addOne(final Other o) {
    o.i++;
  }
}
class Other {
  public int i;
}
```

答案： 没有问题， 虽然o 是final 的， 但是 它的成员变量 i 不是final的， 可以做加一操作

3.7 下面的程序输出什么值？

```
public class JavaTest {
   public static void main(String args[]){
       JavaTest obj=new JavaTest();
         obj.method(100);
       }
       void method(int i){
         System.out.println("int:  "+i);
       }
       void method(long i){
         System.out.println("long:  "+i);
       }
}
```

答案： int : 100

3.8 下面的代码输出什么：

```
class Fruit{
    String name = "Fruit";
    public void print(int i){
        System.out.println("Fruit"+i);
    }
}

class Apple extends Fruit{
    String name = "Apple";
    public void print(int i){
        System.out.println("Apple"+i);
    }
}

public class JavaTest {
    public static void main(String args[]){
        Apple apple = new Apple();
        apple.print(100);
        System.out.println(apple.name);

        Fruit fruit = apple;
        fruit.print(100);
        System.out.println(fruit.name);
    }
}
```

答案：
Apple100
Apple
Apple100
Fruit

3.9 有class A定义如下

```
class A {
 protected int method1(int a, int b) {
  return 0;
 }
}
```

如果有类class B extends A , 在class B中，下列哪两个方法定义是有效的？

A. public int method1(int a, int b) { return 0; }
B. private int method1(int a, int b) { return 0; }
C. private int method1(int a, long b) { return 0; }
D. public short method1(int a, int b) { return 0; }
E. static protected int method1(int a, int b) { return 0; }

答案： A， C
B : 降低了方法可见性
D: 返回值和类A 不兼容
E: 静态方法不能隐藏来自于父类的实例方法

3.10 有一个类OuterClass定义如下：

```
public class OuterClass {
 public void someOuterMethod() {
  // Line 3
 }

 public class InnerClass{
 }
```

```
 public static void main(String[] argv) {
  OuterClass o = new OuterClass();
  // Line 11
 }
}
```

下面哪个语句能够实例化InnerClass?

A. new InnerClass(); // At line 3

B. new InnerClass(); // At line 11

C. new o.InnerClass(); // At line 11

D. new OuterClass.InnerClass(); // At line 11

答案： A

3.11 下面的代码输出什么？

```
abstract class A{
    public abstract void before();
    public abstract void after();
    public void print(int i){
        before();
        System.out.println("A"+i);
        after();
    }
}

class B extends A{
    public  void before(){
        System.out.println("before B");
    }
    public  void after(){
        System.out.println("after B");
    }
}
public class JavaTest {
    public static void main(String args[]){
        A a = new B();
        a.print(100);
    }
}
```

答案：

before B

A100

after B

3.12 下面的代码输出结果是什么？

```
class Fruit {
 public void setDate(Object d) {
  System.out.println("Fruit.setDate(Object d)");
 }
}
class Apple extends Fruit {
 public void setDate(Date d) {
  System.out.println("Apple.setDate(Date d)");
 }
}
public class JavaTest3 {
 public static void main(String[] args) {
  Fruit f = new Apple();
  f.setDate(new Date());
 }
}
```

答案：Fruit.setDate(Object d)

3.13 下列关于类的继承的描述，正确的有哪些？
A. 一个类可以同时继承多个父类
B. 一个类可以具有多个子类
C. 子类会自动拥有父类所有的方法
D. 一个类继承另一个类需要使用 extends 关键字

答案： B, D

3.14 有java 类如下：

```
public class JavaTest {
 public static void main(String args[]) {
  System.out.println(new JavaTest());
 }
}
```

运行该Java 类， 出现的结果是： JavaTest@15db9742
为什么会这样？

答案： System.out.println 默认会调用对象的toString()方法来显示， 但是JavaTest没有定义自己的toString()方法， 于是就调用了父类Object 的toString()方法：

```
public String toString() {
      return getClass().getName() + "@" + Integer.toHexString(hashCode());
}
```

可以看出Object 的对应方法打印了类名和hashCode，注意hashCode()方法在JavaTest中也没有定义， 调用的也是Object 的对应方法，默认值为这个对象在JVM中的id。

3.15 在面向对象程序中，实现代码复用主要有两种方式：继承和组合， 请描述下这两种方式的区别。
要点：
继承：复用父类的属性和方法， 在编译期间就确定了类的层次结构， 在实际编程中， 不好的设计经常会导致父类中带有子类的行为， 导致父类和子类互相依赖， 丧失了重用的好处，破坏了封装。
组合：在运行期间通过对象之间的引用动态定义的之间的关系，对象之间只能通过接口相互作用，对象的封装性也就得到了良好地维护，在运行期间，任何对象都能够被替换为其 他相同类型的对象
总而言之： 优先使用组合，而不是继承。

关于继承一个不好的例子：

```
public class Stack extends Vector{
  ...
}
```

导致Stack类拥有了一些不属于栈的方法，例如 add(int index, Object o); remove(int index)

第四部分： Java 集合

4.1 Java 集合框架的Map, Set和List 有什么区别？
要点：
Map 描述的是 key-value 这样的关系
Set 是一个没有重复元素的集合
List 描述了一个带有顺序的， 可以用索引值(0, 1,3,4.....) 去访问的列表。
4.2 为什么Map 接口不继承Collection接口？
要点： 因为Map接口描述的是 Key-value 结构， Collection 描述了一组元素的集合， 二者之间没有继承的关系。
4.3 为什么要有Iterator?
要点：Iterator 是对一个集合遍历操作的抽象， 它可以屏蔽集合内部数据结构的细节， 只要具备hasNext()，next()方法就可以顺次取出集合中的元素。

4.4 hashCode() 和 equals()方法是如何被HashMap所使用的？
要点：例如把一个自己定义的class Foo{...} 作为key放到HashMap。
实际上HashMap也是把数据存一个类似数组的结构中，所以在put函数里面，HashMap会调Foo.hashCode()算出作为这个元素在数组的位置，然后把key和value封装成一个对象放到数组。
如果有2个对象计算出的hash code 一样， 那就利用Foo.equals()方法 和现有的对象比较， 如果为true, 现有对象被冲掉， 如果为false, 那个HashMap里面的数组的这个元素就变成了链表。 也就是hash code一样的元素在一个链表里面，链表的头在那个数组里面。

4.5 Map接口提供了哪些集合视图可供使用？
答案： keySet, values(), entrySet

4.6 如何决定选用HashMap还是TreeMap？
答案： TreeMap 的key 是有序的

4.7 我们如何从给定集合那里创建一个synchronized的集合？
答案： 调用Collections.synchronizedCollection(....)

4.8 当一个集合被作为参数传递给一个函数时，如何才可以确保函数不能修改它？
答案： 调用Collections.unmodifiableCollection(...)

4.9 下列关于ArrayList、Vector和LinkedList集合的说法正确是有哪些？
A. ArrayList集合底层是数组实现，该集合线程不安全
B. Vector 集合元素的存放是无序的
C. LinkedList集合底层是链表结构，适合做元素的增删操作
D. 这三个集合都是List接口的实现类

答案： A，C, D

4.10 下面程序的输出结果是什么？

```
Set<String> set= new HashSet<String>();
set.add("aaa");
set.add("bbb");
set.add("aaa");
System.out.println(set.size());
```

A. 编译不通过
B. 运行时出错
C. 输出3
D. 输出2

答案： D

4.11 下列哪些集合属于Collection体系的子类？
A. TreeMap
B. ArrayList
C. Hashtable
D. HashSet

答案： B，D

第五部分 Java 异常

5.1 Error和Exception 有什么区别？
答案： Error是程序无法处理的错误，比如OutOfMemoryError、ThreadDeath等。这些异常发生时，Java虚拟机（JVM）一般会选择线程终止。

Exception是程序本身可以处理的异常，这种异常分两大类运行时异常和非运行时异常。程序中应当尽可能去处理这些异常。

5.2 Java中的检查型异常和非检查型异常有什么区别？
要点：在Java中所有不是RuntimeException派生的Exception都是检查型异常。当函数中存在抛出检查型异常的操作时该函数的函数声明中必须包含throws语句。调用改函数的函数也必须对该异常进行处理，如不进行处理则必须在调用函数上声明throws语句。

在Java中所有RuntimeException的派生类都是非检查型异常，非检查型异常可以不在函数声明中添加throws语句，调用函数上也不需要强制处理。

Effective Java 中关于异常的使用准则：
对于调用者不可能从其中恢复的情形，或者惟一可以预见的响应将是程序退出，则不要使用检查型异常。
运行时异常应该只是用于指示编程错误，例如违反前置条件。
一个方法所抛出的异常应该在一个抽象层次上定义，该抽象层次与该方法做什么相一致，而不一定与方法的底层实现细节相一致，例如，一个从文件、数据库或者 JNDI 装载资源的方法在不能找到资源时，应该抛出某种ResourceNotFound 异常（通常使用异常链来保存隐含的原因），而不是更底层的IOException 、SQLException 或者NamingException 。

5.3 finally的作用是什么？ 什么时候会被执行？

要点：无论是否抛出异常，finally代码块总是会被执行。就算是没有catch语句同时又抛出异常的情况下，finally代码块仍然会被执行。最后要说的是，finally代码块主要用来释放资源，比如：I/O缓冲区，数据库连接。

5.4 finally, finalize有什么区别？

要点：
finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。JVM不保证此方法总被调用

下面有关Java异常处理的说法错误的是
A. 一个try块只能有一条catch语句
B. 一个try块中可以不使用catch语句
C. catch块不能单独使用，必须始终与try块在一起
D. finally块不能单独使用，必须始终与try块在一起

答案：A

5.5 下面代码有什么问题？

```
public static void start() throws IOException, RuntimeException{
    throw new RuntimeException("Not able to Start");
}

public static void main(String args[]) {
 try {
  start();
 } catch (Exception ex) {
  ex.printStackTrace();
 } catch (RuntimeException re) {
  re.printStackTrace();
 }
}
```

答案：catch (RuntimeException re)是无法到达的代码块

5.6 把下面4种情况和后面的四种错误做一个匹配

1, int[] a;　　a[0] = 0;
2. JVM正在运行你的程序的时候，突然找不到一个系统相关的类（位于rt.jar/classes.zip中）
3. 一个程序从IO Stream中读取数据，并且到达了Stream的结束处
4. 一个程序从一个已经关闭了的IO Stream中读取数据

a. 出现Error
b. 编译错误
c. 抛出检查型异常
d. 没有错误

答案：1-b　2-a　3-d　4-c

5.7 下面程序有何错误，该如何修改？

```
public static void cat(File file) {
    RandomAccessFile input = null;
    String line = null;

    try {
        input = new RandomAccessFile(file, "r");
        while ((line = input.readLine()) != null) {
            System.out.println(line);
        }
        return;
```

```
    } finally {
        if (input != null) {
            input.close();
        }
    }
}
```

答案：

```
public static void cat(File file) {
    RandomAccessFile input = null;
    String line = null;

    try {
        input = new RandomAccessFile(file, "r");
        while ((line = input.readLine()) != null) {
            System.out.println(line);
        }
        return;
    } catch(FileNotFoundException fnf) {
        System.err.format("File: %s not found%n", file);
    } catch(IOException e) {
        System.err.println(e.toString());
    } finally {
        if (input != null) {
            try {
                input.close();
            } catch(IOException io) {
            }
        }
    }
}
```

第六部分 Java IO

6.1 字节流与字符流有什么区别?

要点：要把一块二进制数据数据输出到某个设备中，或者从某个设备中逐一读取一块二进制数据，不管输入输出设备是什么，我们要用统一的方式来完成这些操作，用一种抽象的方式进行描述，这个抽象描述方式起名为IO流，对应的抽象类为OutputStream和InputStream，不同的实现类就代表不同的输入和输出设备，它们都是针对字节进行操作的。

在应用中，经常要完全是字符的一段文本输出去或读进来，用字节流可以吗？计算机中的一切最终都是二进制的字节形式存在。对于“中国”这些字符，首先要得到其对应的字节，然后将字节写入到输出流。读取时，首先读到的是字节，可是我们要把它显示为字符，我们需要将字节转换成字符。由于这样的需求很广泛，人家专门提供了字符流的包装类。

6.2 下面程序的运行结果是什么？

```
FileOutputStream fos = new FileOutputStream(“c:\\demo.txt”);
fos.write(“abc”);
fos.close();
```

A.在C盘创建文件demo.txt,但文件是空的
B.在C盘创建文件demo.txt,并写入数据abc
C.将C盘已有的文件demo.txt中追加写入abc
D.编译失败

答案：D ，FileOutputStream 字节流，没有一个write 方法，可以写入字符串

6.3 下面代码的作用是什么？

```
BufferedReader br = new BufferedReader(new FileReader("c:\\a.txt"));
BufferedWriter bw = new BufferedWriter(new FileWriter("d:\\b.txt"));
String line = null;
while ((line = br.readLine()) != null) {
 bw.write(line);
 bw.newLine();
 bw.flush();
}
```

bw.close();
br.close();

A. 把c盘目录下的a.txt文件内容复制到d盘目录下的b.txt
B. 把d盘目录下的b.txt文件内容复制到c盘目录下的a.txt
C. 读取c盘目录下a.txt文件，输出在控制台
D. 把控制台的内容写入到d盘目录下的b.txt文件中

答案： A

6.4 （本题7分）有一个二进制文件shoppingcart.data, 其中存储的数据依次是：
产品价格(double类型)， 产品数量(int类型 ）， 产品描述(UTF字符串)
这样的数据可能有多组， 写一个程序，读取该二进制文件的内容， 计算出该购物车的总价格(即每一项 产品价格*产品数量 的和)

提示：使用DataInputStream
参考答案：

```
private static void readData() throws IOException {
  DataInputStream in = null;
  double total = 0.0;
  try {
   in = new DataInputStream(new BufferedInputStream(new
                            FileInputStream(dataFile)));

   double price;
   int unit;
   String desc;

   try {
    while (true) {
     price = in.readDouble();
     unit = in.readInt();
     desc = in.readUTF();
     total += unit * price;
    }
   } catch (EOFException e) {
   }
   System.out.format("TOTAL price: $%.2f%n", total);
  } finally {
   in.close();
  }
 }
```

6.5 （本题7分）写一个程序， 读取一个.class文件的前4个字节， 转换成十六进制字符，检查是不是：CAFEBABE

参考答案：

第七部分 Java 线程

7.1 为什么Java中有多线程编程， 而很少提到Java多进程编程？

要点： 因为JVM (java.exe) 本身就是一个进程， 在其中只能说线程编程了

7.2 创建一个线程有那些方法？
要点：
继承Thread类
实现Runnable接口

7.3 为什么我们调用start()方法时会执行run()方法，为什么我们不能直接调用run()方法？

参考答案：通过调用Thread类的 start()方法来启动一个线程，这时此线程处于就绪（可运行）状态，并没有运行，一旦得到cpu时间片，就开始执行run()方法，这里方法 run()称为线程体，它包含了要执行的这个线程的内容，Run方法运行结束，此线程随即终止

而run方法只是thread的一个普通方法调用，如果直接调用它，还是在主线程里执行

7.4 Java 中多线程之间在什么情况下需要同步？

要点：当多个线程并发的操作一个可以共享的资源的时候

7.5 什么是不可变对象，它对写并发应用有什么帮助？
要点：不可变对象（英语：Immutable object）是一种对象,在被创造之后，它的状态就不可以被改变。任何试图改变改对象状态的操作都会生成新的对象，老的对象保持不变。

在多线程并发的情况下，多个线程读取共享变量是安全的，但是写操作肯定会冲突，由于不可变对象不可更改，并发时不需要其他额外的同步保证，故相比其他的锁同步等方式的并发性能要好。

7.6 现在有t1、t2、t3三个线程，你怎样保证t2在t1执行完后执行，t3在t2执行完后执行？
答案：在t3线程中，调用t2.join，在t2中，调用t1.join

7.7 有t1, t2, t3 三个线程，都要对 r1, r2, r3, r4 这4个资源进行读写，对每个资源读写之前需要获得锁才能操作，由于线程的乱序执行，可能导致死锁。
请设计一种策略来避免死锁的产生。

要点：无论是那个线程，只要对这4个资源访问，一定要按照同样的次序来申请锁，例如r1, r2, r3, r4，或者 r4, r3, r2, r1，只要大家都按同样的次序来申请锁就没问题。

第八部分 Java 反射

8.1 为什么需要反射？

要点：反射指的是在运行时根据名字获取一个类的属性和方法信息，创建对象，调用方法的一种机制。
由于这种动态性，可以极大增加程序的灵活性，程序不用在编译期就完全确定，在运行期仍然可以扩展。
尤其在一些框架中，框架的作者事先并不知道使用者要创建什么对象，所以由使用者告诉框架类的名称，方法的名称，然后框架通过反射来创建对象，调用方法。

8.2 获取Class类型的对象有哪些：
A. Object类的getClass()
B. class静态属性
C. 自己创建Class对象
D. Class类的forName()静态方法

答案：A, B, D

8.3（本题7分）有Java 类如下：

```
public class Employee {

 private String id;
 private String name;
 private int age;
 private Employee(){
 }
 public Employee(String name, int age) {
  this.id = "1001";
  this.name = name;
  this.age = age;
 }
 private String getID(){
  return this.id;
 }
 public void sayHello(){
  System.out.println("Hello, name ="+ name + "  age = "+age);
 }
}
```

要求：

(1) 通过反射的方式创建改Employee的实例
(2) 通过反射调用sayHello()方法
(3) 通过反射调用getID()方法
(4) 打印出每个字段，格式为
权限描述符(private ,protected 等) + 字段类型+字段名称

参考答案：

```
Class<?> clz = Class.forName("Employee");

Constructor<?> c = clz.getConstructor(String.class,int.class);

Object obj = c.newInstance("andy",23);

Method m = clz.getDeclaredMethod("sayHello");
m.invoke(obj);

m = clz.getDeclaredMethod("getID");
m.setAccessible(true);
String id = (String)m.invoke(obj);
System.out.println(id);


Field[] field = clz.getDeclaredFields();
for (int i = 0; i < field.length; i++) {
 // 权限修饰符
 int mo = field[i].getModifiers();
 String priv = Modifier.toString(mo);
 // 属性类型
 Class<?> type = field[i].getType();
 System.out.println(priv + " " + type.getName() + " " + field[i].getName() + ";");
}
```

8.4 有一个支持Integer 的ArrayList如下：

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

通过反射向该ArrayList中加入一个String类型的元素

参考答案：

```
ArrayList<Integer> list = new ArrayList<Integer>();
Method method = list.getClass().getMethod("add", Object.class);
method.invoke(list, "hello");
System.out.println(list.get(0));
```

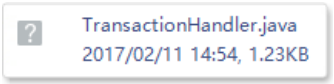
8.5 （本题8分）有AccountService, AccountServiceImpl , Transaction 类如下：

```
public interface AccountService {
 public void transfer(String fromAccount,
     String toAccount,
     int ammount);
   public void query(String accountId);
}
public class AccountServiceImpl implements AccountService {

 @Override
 public void transfer(String fromAccount, String toAccount, int ammount) {
  System.out.println("Transfer " + ammount + " from " + fromAccount + " to  " + toAccount );
```

```
 }

 @Override
 public void query(String accountId){
  System.out.println("query account id:" +accountId);
 }
}
public class Transaction {
 public void beginTx(){
  System.out.println("Begin transaction");
 }
 public void commitTx(){
  System.out.println("commit transaction");
 }
}
```

试用Java 动态代理实现如下功能：
在调用AccountServiceImpl 的transfer("A100", "B200", 233) 方法时，输出如下：
Begin transaction
Transfer 233 from A100 to B200
commit transaction
在调用AccountServiceImpl 的query("A100")，输出结果：
query account id:A100

参考答案：



第九部分 Java 泛型

9.1 下列说法错误的是

A . Java中的泛型基本上都是在编译器这个层次来实现的。
B. 在生成的Java字节码中是不包含泛型中的类型信息的。
C. 泛型可以应用在接口，类，和方法上。
D. ArrayList<int> list = new ArrayList<int> 是合法的

答案：D

9.2 下面的代码是否正确

```
public class AnimalHouse<E> {
 private E animal;
 AnimalHouse(){
  animal = new E();
 }
}
```

答案：错误，不能够new E

9.3 下面的代码输出的结果是什么？

```
ArrayList<String> list1 = new ArrayList<String>();
ArrayList<Integer> list2 = new ArrayList<Integer>();
System.out.println(list1.getClass().equals(list2.getClass()));
```

答案：true

9.4 下面的代码有问题吗？ 为什么？

```
ArrayList<Number> numbers = new ArrayList<Number>();
numbers.add(new Integer(10));
numbers.add(new Double(10.0d));
```

答案：没有问题，因为Integer, Double 都是Number的子类，ArrayList<E> 的add(E e) 方法经过编译器进行类型擦除以后变成了add(Number e), 可以加入Integer 和Double

9.5 下面的泛型类经过类型擦除以后会变成什么？

```
public class AnimalHouse<T> {
 private T animal;
 public T getAnimal(){
  return this.animal;
 }
}
```

答案：

```
public class AnimalHouse {
 private Object animal;
 public Object getAnimal(){
  return this.animal;
 }
}
```

9.6 泛型类同上一题， 有如下代码：

```
AnimalHouse<Tiger> house = new AnimalHouse<Tiger>;
Tiger t = house.getAnimal();
```

经过类型擦除，在执行getAnimal()是JVM会怎么做？

答案：

先返回一个Object 类型的数据，然后强制转型为 Tiger

```
Tiger t = (Tiger) getAnimal();
```

9.7 下面的代码有什么问题？ 该如何改正：

```
public static <T> T  compare(T t1,T t2){
 if(t1.compareTo(t2) >= 0){
  return t1;
 } else{
  return t2;
 }
}
```

答案：会有编译错误，因为在编译期间，并不知道 T的类型，自然不能调用compareTo方法。

改正方法：添加类型限定

```
public static <T extends Comparable> T compare(T t1,T t2){
 if(t1.compareTo(t2) >= 0){
  return t1;
 } else{
  return t2;
 }
}
```

9.8 下面代码的问题是什么？ 该如何改正？

```
public static void main(String[] args) {
 ArrayList<Integer> list = new ArrayList<Integer>();
 show(list);
}
public static void show(ArrayList<Number> list){
 //....
}
```

答案：有问题，因为ArrayList<Integer>并不是ArrayList<Number>的子类型。

```
public static void main(String[] args) {
 ArrayList<Integer> list = new ArrayList<Integer>();
 show(list);
}
public static void show(ArrayList<? extends Number> list){
  //....
}
```