

JAVA应用技术

复习课

知识点 I

- Java 常识
 - Java 代码编译执行的过程
 - 真实编译
 - 字节码解释执行
 - 跨平台的原因
 - JVM
 - 数据类型统一
 - Java 和 C++ 的比较
 - Java 内存模型
 - 对象都在堆里
 - 对象变量是指针
 - 垃圾回收机制
 - 数组下标检查
 - 单根结构：Object 类
 - main()
 - `public static void main(String[] args)`
 - 命令行参数
 - Java 关键字

- Java 基础：数据类型、对象和控制语句
 - 基本数据类型
 - 对象变量
 - 对象变量的意义
 - 对象变量的赋值
 - 对象变量做函数参数和返回值
 - 对象变量的比较
 - equals()
 - 字符串的连接
 - ?:运算符的结果类型问题
 - 带标号的 `break` 和 `continue`
- 类
 - `this`
 - 在成员函数内使用
 - 调用其他构造函数
 - 成员变量初始化
 - 定义初始化
 - 构造函数初始化
 - 静态成员
 - 静态成员的访问：通过“.”运算符
 - 静态成员变量和类对象的关系

知识点 II

- 数组
 - 数组的创建
 - 数组变量的赋值
 - 对象数组
 - for-each 循环
 - 对象数组 for-each 的特殊性
- 访问属性
 - import 的意义
 - package 和 CLASSPATH
 - 默认访问属性：包内
 - protected：子类及包内
 - class 的访问属性
 - 默认类仅限包内访问
 - public 类必须和源代码文件同名
- 继承和多态
 - 单继承
 - super 的作用
 - 继承和私有变量的关系
 - 和 C++ 的两个区别
 - 构造函数内实现了动态绑定
 - 没有名字隐藏
 - 默认动态绑定
 - final
 - final 变量
 - final 函数和类

- 特殊的类
 - 抽象
 - 接口
 - 接口作为数据类型
 - 接口可以多继承
 - 接口内的 default 函数
 - 内部类
 - 定义
 - 和外部类的关系
 - 匿名类的语法
 - 枚举类
 - 定义
 - 构造函数和成员函数
- 容器
 - 主要容器类型
 - List
 - Set
 - Map
 - 容器实现方式
 - ArrayList vs LinkedList
 - HashMap vs TreeMap
 - 遍历
 - Iterator
 - for-each
 - 范型的使用
 - 子类型范型和通配符

知识点 III

- 标准类库
 - Object类
 - equals
 - clone
 - 包裹类
 - 自动打包/解包
 - 包裹类的比较
 - 常量对象
 - Math类
 - String类
 - 理解String是不可写的对象
 - 常用函数
 - 在switch-case中使用
 - StringBuffer类
 - Random类
- 异常
 - throw-try-catch机制
 - throw
 - catch的匹配方式
 - 万能catch
 - 再抛出
 - Throwable接口的方法
 - finally
 - 函数对抛出异常的声明throws
 - 编译时检查
 - 与构造函数的关系
 - 与函数覆盖的关系

- IO
 - stream
 - 流的基本概念
 - 输入输出分开
 - 只处理byte
 - 流的基本函数
 - 文件流的使用
 - Reader/Writer和stream的关系
 - 通过桥建立两者的关系
 - 如何做汉字编码转换
 - DataInput/OutputStream
 - 理解二进制流
 - 对象串行化

- GUI
 - AWT和Swing
 - 部件、容器、布局管理器的关系
 - JFrame类的使用
 - add()
 - pack()
 - setDefaultCloseOperation()
 - Graphics类的使用
 - 理解paint()函数
 - 常见布局管理器的效用
 - 菜单类族的使用
 - Swing的消息机制
 - 消息机制
 - Listener、Event类
 - add/removeListener函数
 - 理解以线程方式通知
 - 常见部件（略）
 - JTable与MVC模式

知识点 IV

- 线程
 - 创建线程
 - Runnable接口
 - Thread类
 - 线程控制
 - start()
 - sleep()
 - yield()
 - 优先级控制
 - 线程同步: synchronized
 - 线程的wait()和notify()机制
 - 通过管道的线程间通信
- RTTI
 - Class类
 - getClass()
 - .class
 - isInstance()
 - 从Class类对象中获得父类、接口和函数的方法
 - Method类
 - invoke()
 - instanceof运算符
- socket通信
 - TCP的Socket和ServerSocket
 - UDP的通信方式
 - 构建socket服务的设计模式

- JDBC
 - SQLite
 - JDBC如何连接和查询
 - 事务处理
 - preparedStatement
- 函数式编程
 - Lambda表达式
 - 函数式接口
- 流式计算
 - 容器的stream接口
 - 常用的高阶函数
 - 过滤
 - 映射
 - 聚合

样题

判断题

`char` of Java is 8-bit. (1分)

F

A Java class can extend from multiple base classes. (1分)

F

Member variables are to get default init values when the object is to be created. (1分)

T

`protected` member can be visited by extended class only. (1分)

F

InputStream and OutputStream read and write 8-bit data. (1分)

T

Swing container is used to organize other GUI components in. But other containers can not be put in a container. (1分)

F

To access a method of a class, an object of that class must be created first. (1分)

F

When an object is de-serialized, its constructor does not run. (1分)

T

样题

单选题

About Java containers, which statement below is NOT correct? (2分)

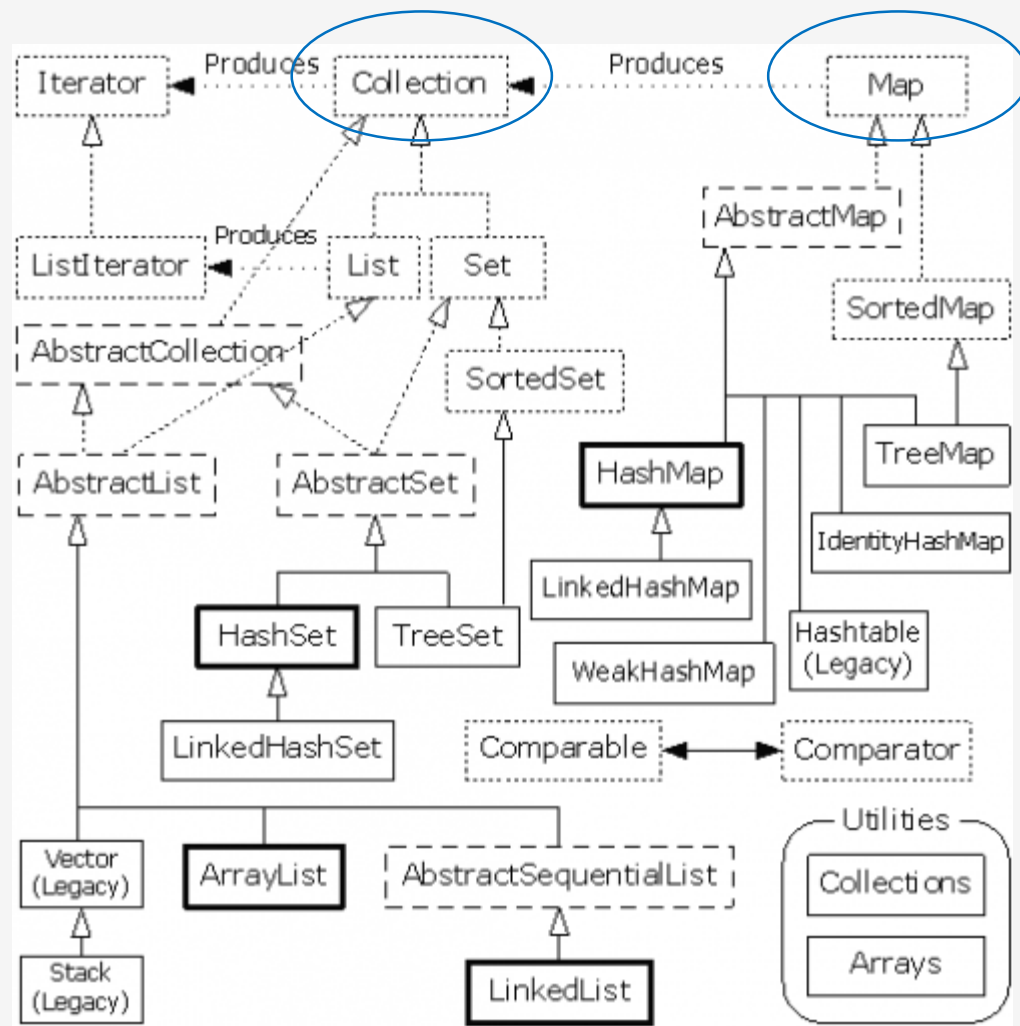
- ☐ A. `List` holds the elements in a particular sequence
- ☐ B. `Set` cannot have any duplicate elements
- ☐ C. `Map` has group of key-value object pairs
- ☒ D. `Iterator` can deal with `List`, `Set` and `Map`

HashMap

```
public Set<K> keySet() {  
    Set<K> ks = keySet;  
    if (ks == null) {  
        ks = new KeySet();  
        keySet = ks;  
    }  
    return ks;  
}
```

```
public Collection<V> values() {  
    Collection<V> vs = values;  
    if (vs == null) {  
        vs = new Values();  
        values = vs;  
    }  
    return vs;  
}
```

```
public Set<Map.Entry<K,V>> entrySet() {  
    Set<Map.Entry<K,V>> es;  
    return (es = entrySet) == null ? (entrySet = new EntrySet()) : es;  
}
```



样题

For swing event handling mechanism, which one below is NOT correct? (2分)

- ☐ A. Event source like JButton is able to have more than one ActionListener objects registered
- ☐ B. When an event occurs, the source object notices all the registered listeners
- ☐ C. A registered listener is able to be de-registered from a source object dynamically
- ☒ D. One listener can not be registered at more than one source object

For code below:

```
ArrayList<Integer> a = new ArrayList<Integer>();  
ArrayList<Double> b = new ArrayList<Double>();
```

Which statement below is **NOT** correct? (2分)

- ☐ A. a.getClass().equals(b.getClass()) is true
- ☐ B. a.getClass() == b.getClass() is true
- ☐ C. a instanceof ArrayList is true
- ☒ D. a.getClass() == b.getClass() is false

样题

Given code below:

```
class Base{
    public final void method() {
        System.out.println("Base.method");
    }
}

public final class Fin extends Base{
    public void method() {
        System.out.println("Fin.method");
    }
    public static void main(String argv[]){
        Base b = new Fin();
        b.method();
    }
}
```

final方法不能被重写。

While one below is correct? (2分)

- ☐ A. It does not compile because of method() in Fin is not defined final as its base one
- ☐ B. It does not compile because Fin can not be final
- ☒ C. It does not compile because of method() in Base final so no function can override it in derived classes
- ☐ D. It compiles and prints Fin.method

样题

For `InputStream.read()`, the `read()` with no parameters, which statement below is correct? (2分)

- ☒ A. `read()` returns `int`, because it has to return EOF to indicate the end of the file
- ☐ B. `read()` returns `byte`, because it reads a byte from the stream
- ☐ C. `read()` returns `char`, because it reads a `char` from the stream
- ☐ D. `read()` returns `int`, as the number of bytes it just read

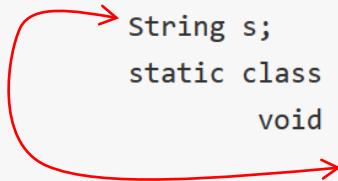
The value returned is a byte as an `int` type.

| <i>java.io.InputStream</i> | |
|--|--|
| <code>+read(): int</code> | Reads the next byte of data from the input stream. The value byte is returned as an <code>int</code> value in the range <u>0 to 255</u> . If no byte is available because the end of the stream has been reached, the value <code>-1</code> is returned. |
| <code>+read(b: byte[]): int</code> | Reads up to <code>b.length</code> bytes into array <code>b</code> from the input stream and returns the actual number of bytes read. Returns <code>-1</code> at the end of the stream. |
| <code>+read(b: byte[], off: int, len: int): int</code> | Reads bytes from the input stream and stores into <code>b[off]</code> , <code>b[off+1]</code> , ..., <code>b[off+len-1]</code> . The actual number of bytes read is returned. Returns <code>-1</code> at the end of the stream. |
| <code>+available(): int</code> | Returns the number of bytes that can be read from the input stream. |
| <code>+close(): void</code> | Closes this input stream and releases any system resources associated with the stream. |
| <code>+skip(n: long): long</code> | Skips over and discards <code>n</code> bytes of data from this input stream. The actual number of bytes skipped is returned. |
| <code>+markSupported(): boolean</code> | Tests if this input stream supports the <code>mark</code> and <code>reset</code> methods. |
| <code>+mark(readlimit: int): void</code> | Marks the current position in this input stream. |
| <code>+reset(): void</code> | Repositions this stream to the position at the time the <code>mark</code> method was last called on this input stream. |

样题

Given the following code:

```
public class Test {  
    String s;  
    static class Inner {  
        void testMethod() {  
            s = "Hello world.";  
        }  
    }  
    public static void main(String[] argv) {  
        Inner i = new Inner();  
        i.testMethod();  
        System.out.println(s);  
    }  
}
```



- Since a static inner class has no connection to an object of the outer class, within an inner class method
 - Instance variables of the outer class **cannot be** referenced
 - Nonstatic methods of the outer class **cannot be** invoked

which one below is correct?(2分)

- ☐ A. It compiles and prints out Hello world.
- ☒ B. It does not compile because String s in class Test is not static.
- ☐ C. It does not compile because Inner can not used in the way in main()
- ☐ D. It compiles and exception raises at running indicates that s has not been initiated.

样题

Given the following class definition which of the following can be legally placed after the comment line `//Here`?(2分)

```
class Base{
    public Base(int i){}
}

public class MyOver extends Base{
    public static void main(String arg[]){
        MyOver m = new MyOver(10);
    }
    MyOver(int i){
        super(i);
    }
    MyOver(String s, int i){
        this(i);
        //Here
    }
}
```

- ☐ A. `MyOver m = new MyOver();`
- ☐ B. `super();`
- ☐ C. `this("Hello",10);`
- ☒ D. `Base b = new Base(10);`

样题

Why might you define a method as native? (2分)

- ☒ A. To get to access hardware that Java does not know about
- ☐ B. To define a new data type such as an unsigned integer
- ☐ C. To write optimized code for performance in a language such as C/C++
- ☐ D. To overcome the limitation of the private scope of a method

For exception, which statement below is **NOT** correct? (2分)

- ☐ A. It is possible to have a try block with out any catch clause but a finally clause
- ☐ B. It is possible to have a try block inside another try block
- ☐ C. It is possible to have a try block along with its catch clauses inside a catch clause
- ☒ D. To re-throw the exception object in a catch clause, simple put a throw statement without the name of the object.

Which of the following will output -3.0 (2分)

- ☐ A. `System.out.println(Math.floor(-3.7));`
- ☐ B. `System.out.println(Math.round(-3.7));`
- ☒ C. `System.out.println(Math.ceil(-3.7));`
- ☐ D. `System.out.println(Math.min(-3.7));`

样题

What must be done when throwing an integer as an exception? (2分)

- ☒ A. Integers cannot be thrown.
- ☐ B. Declare integers as Throwable.
- ☐ C. Import the exception class.
- ☐ D. Encapsulate the integer handler

What best describes the appearance of an application with the following code?

```
public class FlowAp extends Frame{
public static void main(String argv[]){
    FlowAp fa=new FlowAp();
    fa.setSize(400,300);
    fa.setVisible(true);

}

FlowAp(){
    add(new Button("One"));
    add(new Button("Two"));
    add(new Button("Three"));
    add(new Button("Four"));
}
}
```

(2分)

- ☐ A. A Frame with buttons marked One to Four placed on each edge.
- ☐ B. A Frame with buttons marked One to four running from the top to bottom
- ☒ C. A Frame with one large button marked Four in the Centre
- ☐ D. An Error at run time indicating you have not set a LayoutManager

样题

What will happen when you attempt to compile and run the following code?

```
public class Bground extends Thread{
    public static void main(String argv[]){
        Bground b = new Bground();
        b.run();
    }
    public void start(){
        for (int i = 0; i <10; i++){
            System.out.println("Value of i = " + i);
        }
    }
}
```

(2分)

- ☐ A. A compile time error indicating that no run method is defined for the Thread class
- ☐ B. A run time error indicating that no run method is defined for the Thread class
- ☐ C. Clean compile and at run time the values 0 to 9 are printed out
- ☒ D. Clean compile but no output at runtime

样题

Suppose there is no file Hello.txt in the current directory. Run the program: (2分)

```
import java.io.*;
public class ABC {
    public static void main(String argv[]) throws Exception {
        ABC m=new ABC();
        System.out.println(m.ff());
    }

    public int ff() {
        try {
            FileInputStream dis=new FileInputStream("Hello.txt");
        } catch (FileNotFoundException fne) {
            System.out.print("No such file found, ");
            throw fne;
        } finally {
            System.out.print("Doing finally, ");
        }
        return 0;
    }
}
```

- ☐ A. No such file found,
- ☐ B. No such file found ,0
- ☒ C. No such file found, Doing finally,
- ☐ D. No such file found, Doing finally, 0

样题

About layout manager in AWT and Swing, which one below is correct? (2分)

- ☐ A. `FlowLayout` is the default layout manager of `Frame`.
- ☒ B. `GridLayout` divides the whole space into even pieces.
- ☐ C. It is not possible to specify coordinates of component regardless the effect of any layout managers.
- ☐ D. Every place in a `BorderLayout` has to be fill with a component, or it will leave blank.

Which statement below is NOT correct? (2分)

- ☐ A. A thread is an instance of Thread class.
- ☐ B. A thread runs the run() method of the Runnable object.
- ☒ C. A new born thread can run immediately when start() is called.
- ☐ D. Thread can access data of the Runnable object.

Given code below:

```
List<String> ls = new ArrayList<String>();  
List<Object> lo = ls;  
lo.add(new Object());  
String s = ls.get(0);
```

Which statement below is correct? (2分)

- ☒ A. It does not compile
- ☐ B. It compiles but exception raises at line 2
- ☐ C. It compiles but exception raises at line 3
- ☐ D. It compiles but exception raises at line 4

样题

程序输出题

请写出以下程序运行结果：

```
public class X {  
    public static void main(String [] args) {  
        try {  
            badMethod();  
            System.out.print("A");  
        } catch (RuntimeException ex) {  
            System.out.print("B");  
        } catch (Exception ex1) {  
            System.out.print("C");  
        } finally {  
            System.out.print("D");  
        }  
        System.out.print("E");  
    }  
    public static void badMethod() {  
        throw new RuntimeException();  
    }  
}
```

BDE

(2分)

请写出以下程序运行结果：

```
class Test {  
    public static void main(String[] args) {  
        Integer a = new Integer(3);  
        Integer b = 3;  
        int c = 3;  
        System.out.println(a == b);  
        System.out.println(a == c);  
    }  
}
```

false

(2分)

true

(2分)

请写出以下程序运行结果：

```
public class Test {
    public static void main(String[] args){
        House house1 = new Test().new House(1,100);
        House house2 = (House)house1.clone();
        System.out.println(house1==house2);
        System.out.println(house1.equals(house2));
        System.out.println(house1.whenBuilt==house2.whenBuilt);
        System.out.println(house1.whenBuilt.equals(house2.whenBuilt));
    }
    public class House implements Cloneable, Comparable<House> {
        private int id;
        private int area;
        private java.util.Date whenBuilt;
        public House(int id, int area) {
            this.id = id;
            this.area = area;
            whenBuilt = new java.util.Date();
        }
        @Override
        public Object clone() {
            try {
                House houseClone = (House)super.clone();
                houseClone.whenBuilt = (java.util.Date) (whenBuilt.clone());
                return houseClone;
            } catch (CloneNotSupportedException ex) {
                return null;
            }
        }
        @Override
        public int compareTo(House o) {
            if (area > o.area)
                return 1;
            else if (area < o.area)
                return -1;
            else
                return 0;
        }
    }
}
```

false (2分)

false (2分)

false (2分)

true (2分)

equals没有重写

样题

请写出以下程序运行结果：

```
enum EnumTry {  
    MON, TUE, WED, THU, FRI;  
    public static void main(String[] args) {  
        for (EnumTry e : EnumTry.values()) {  
            System.out.println(  
                e + ":" + e.toString() + ":" + e.ordinal() + ":" + e.name());  
        }  
    }  
}
```

MON:MON:0:MON (2分)

TUE:TUE:1:TUE (2分)

WED:WED:2:WED (2分)

THU:THU:3:THU (2分)

FRI:FRI:4:FRI (2分)

样题

给出以下代码：

```
public class Test {  
    public int t=4;  
    public static void main(String[] args) {  
        new Test().NumberPlay();  
    }  
    public void NumberPlay() {  
        int t=2;  
        t = t+5;  
        this.t = this.t-2;  
        t = t-this.t;  
        System.out.println(t+this.t+"ok");  
    }  
}
```

程序运行后输出结果为：

(2分)

请写出以下程序运行结果：

```
class Main {  
    public static void main(String[] args) {  
        String s1 = "Zhejiang University";  
        String s2 = s1.substring(0, 7);  
        s2.toUpperCase();  
        System.out.println(s2+s1.substring(8));  
    }  
}
```

(2分)

样题

请写出以下程序运行结果：

```
public class Test {  
    public static void main(String[] args) throws Exception{  
        String str = "hello";  
        Method m = str.getClass().getMethod("toUpperCase");  
        System.out.println(m.invoke(str));  
        System.out.println(str);  
    }  
}
```

HELLO (2分)

hello (2分)

```
public enum Main {  
    PLUS    { int eval(int x, int y) { return x + y; } },  
    MINUS   { int eval(int x, int y) { return x - y; } },  
    TIMES   { int eval(int x, int y) { return x * y; } },  
    DIVIDE  { int eval(int x, int y) { return x / y; } };  
    abstract int eval(int x, int y);  
    public static void main(String args[]) {  
        int x = 4;  
        int y = 2;  
        for (Main op : Main.values())  
            System.out.printf("%d %s %d = %d\n", x, op, y, op.eval(x, y));  
    }  
}
```

程序运行结果为（一行一空）：

 (2分) (2分) (2分) (2分)

程序填空题

输入 add 5 3
打印 8

```
import java.util.Scanner;
enum IntOp {
    [      ]
    { // for operator add
        int eval(int i1, int i2) {
            return [      ];
        }
    },

    [      ]
    { // for operator sub
        int eval(int i1, int i2) {
            return [      ];
        }
    };

    int eval(int i1, int i2) {
        return 0;
    };
}

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String op = in.next();
        int i1 = in.nextInt();
        int i2 = in.nextInt();
        System.out.println(IntOp.[      ].eval(i1, i2));
        in.close();
    }
}
```

```
enum IntOp {
    add
    { // for operator add
        int eval(int i1, int i2) {
            return i1+i2;
        }
    },
    sub
    { // for operator sub
        int eval(int i1, int i2) {
            return i1-i2;
        }
    };

    int eval(int i1, int i2) {
        return 0;
    };
}

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String op = in.next();
        int i1 = in.nextInt();
        int i2 = in.nextInt();
        System.out.println(IntOp.valueOf(op).eval(i1, i2));
        in.close();
    }
}
```


程序填空题

输入：

```
Taylor Swift U.S.  
the Beatles U.K.  
the Dynasty Tang China  
the May Flower China
```

统计：以“the”开始的行的国家

```
China  
U.K.
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner in = new Scanner(System.in);  
        ArrayList<String[]> lst = new ArrayList<String[]>();  
        while ( in.hasNextLine() ) {  
            String line = in.nextLine();  
            lst.add(line.split(" "));  
        }  
        Set<String> ret = lst.stream()  
            .filter([                ])  
            .map([                ])  
            .collect(Collectors.toList());  
        ArrayList<String> r = new ArrayList<String>(ret);  
        Collections.sort(r);  
        for ( String s: r ) {  
            System.out.println(s);  
        }  
        in.close();  
    }  
}
```

```
public class Main1 {  
    public static void main(String[] args) {  
        Scanner in = new Scanner(System.in);  
        ArrayList<String[]> lst = new ArrayList<String[]>();  
        while (in.hasNextLine()) {  
            String line = in.nextLine();  
            if (line.equals(""))  
                break;  
            lst.add(line.split(" "));  
        }  
        Set<String> ret = lst.stream().filter(x->x[0].equals("the")).map(x->x[x.length-1])  
            .collect(Collectors.toSet());  
        ArrayList<String> r = new ArrayList<String>(ret);  
        Collections.sort(r);  
        for (String s: r) {  
            System.out.println(s);  
        }  
        in.close();  
    }  
}
```

函数题

题目描述

有一连串任务，需要两个线程交替执行。线程1执行完任务1后，线程2才能执行任务2，接下来线程1执行任务1，如此交替执行下去。直到所有任务执行完毕。（15分）

定义 `Repo` 类代表任务仓库，使用字符串代表任务。该类拥有：

构造函数：

```
/*将传递进来的字符串以空格分隔分解为多个不同的任务，并存储起来。如"1 2 3 4 5 6"被分解成6个任务1,2,3,4,5,6*/
public Repo(String items) {
}
```

方法：

```
int getSize(); //返回Repo包含的任务数量。注意：完成任务的时候，需要将任务删除。
//其他完成任务的方法
```

定义 `Worker1` 与 `Worker2` 类，代表两个交替完成任务的类，可以从Repo对象中获取任务。

###main函数如下：

```
public class Main {
    public static void main(String[] args) throws InterruptedException {
        Scanner sc = new Scanner(System.in);
        Repo repo = new Repo(sc.nextLine());
        Thread t1 = new Thread(new Worker1(repo));
        Thread t2 = new Thread(new Worker2(repo));
        t1.start();
        Thread.yield();
        t2.start();
        sc.close();
    }
}
```

输入样例

```
1 2 3 4 5 6 7 8 9
```

输出样例

```
Thread-0 finish 1
Thread-1 finish 2
Thread-0 finish 3
Thread-1 finish 4
Thread-0 finish 5
Thread-1 finish 6
Thread-0 finish 7
Thread-1 finish 8
Thread-0 finish 9
```

裁判测试程序：

```
/*Repo代码*/

/*Worker1代码*/

/*Worker2代码*/

/*系统已有代码，无需关注*/
```

函数题

题目描述

This program reads a line of logical expression with one logical operator and two boolean values, and evaluates the result. A logical expression is like: (5分)

```
true and false
```

The result of the expression above is: `false`.

The `Main` class and a skeleton of enum `LogicalOp` are provided.

裁判测试程序样例：

函数接口定义：

```
enum LogicalOp {  
  
    boolean test(boolean p1, boolean p2) {  
        return false;  
    };  
}
```

Your `LogicalOp` should provide `and` and `or`.

```
public class Main {  
    public static void main(String[] args) {  
        Scanner in = new Scanner(System.in);  
        boolean p1 = in.nextBoolean();  
        String op = in.next();  
        boolean p2 = in.nextBoolean();  
        System.out.println(LogicalOp.valueOf(op).test(p1, p2));  
        in.close();  
    }  
}  
/* 请在这里填写答案 */
```

输入样例：

```
true and false
```

输出样例：

```
false
```

编程题

题目描述

定义 `IllegalScoreException` 异常类，代表分数相加后超出合理范围的异常。该异常是 `checked exception`，即希望该异常一定要被捕获处理。

定义 `IllegalNameException` 异常类，代表名字设置不合理的异常。该异常是 `unchecked exception`

定义 `Student` 类。

属性:

```
private String name;
private int score;
```

方法:

```
toString          //自动生成
setter/getter      //自动生成
改造setName        //如果姓名首字母为数字则抛出IllegalNameException
public int addScore(int score) //如果加分后分数<0 或>100，则抛出IllegalScoreException，加分不成功。
```

###main方法:

1. 输入 `new` 则新建学生对象。然后输入一行学生数据，格式为 `姓名 年龄`，接着调用`setName`，`addScore`。否则跳出循环。
2. `setName`不成功则抛出异常，并打印异常信息，然后继续下一行的处理。
3. `addScore`不成功则抛出异常，并打印异常信息，然后继续下一行的处理。如果2、3都成功，则打印学生信息(`toString`)
4. 如果在解析学生数据行的时候发生其他异常，则打印异常信息，然后继续下一行的处理。
5. `Scanner`也是一种资源，希望程序中不管有没有抛出异常，都要关闭。关闭后，使用 `System.out.println("scanner closed")` 打印关闭信息

注意：使用 `System.out.println(e)`；打印异常信息，e为所产生的异常。

输入样例：

```
new
zhang 10
new
wang 101
new
wang30
new
3a 100
new
wang 50
other
```

输出样例：

```
Student [name=zhang, score=10]
IllegalScoreException: score out of range, score=101
java.util.NoSuchElementException
IllegalNameException: the first char of name must not be digit, name=3a
Student [name=wang, score=50]
scanner closed
```