考试模式：开卷
试卷结构（A卷）

一、判断题

二、程序阅读题

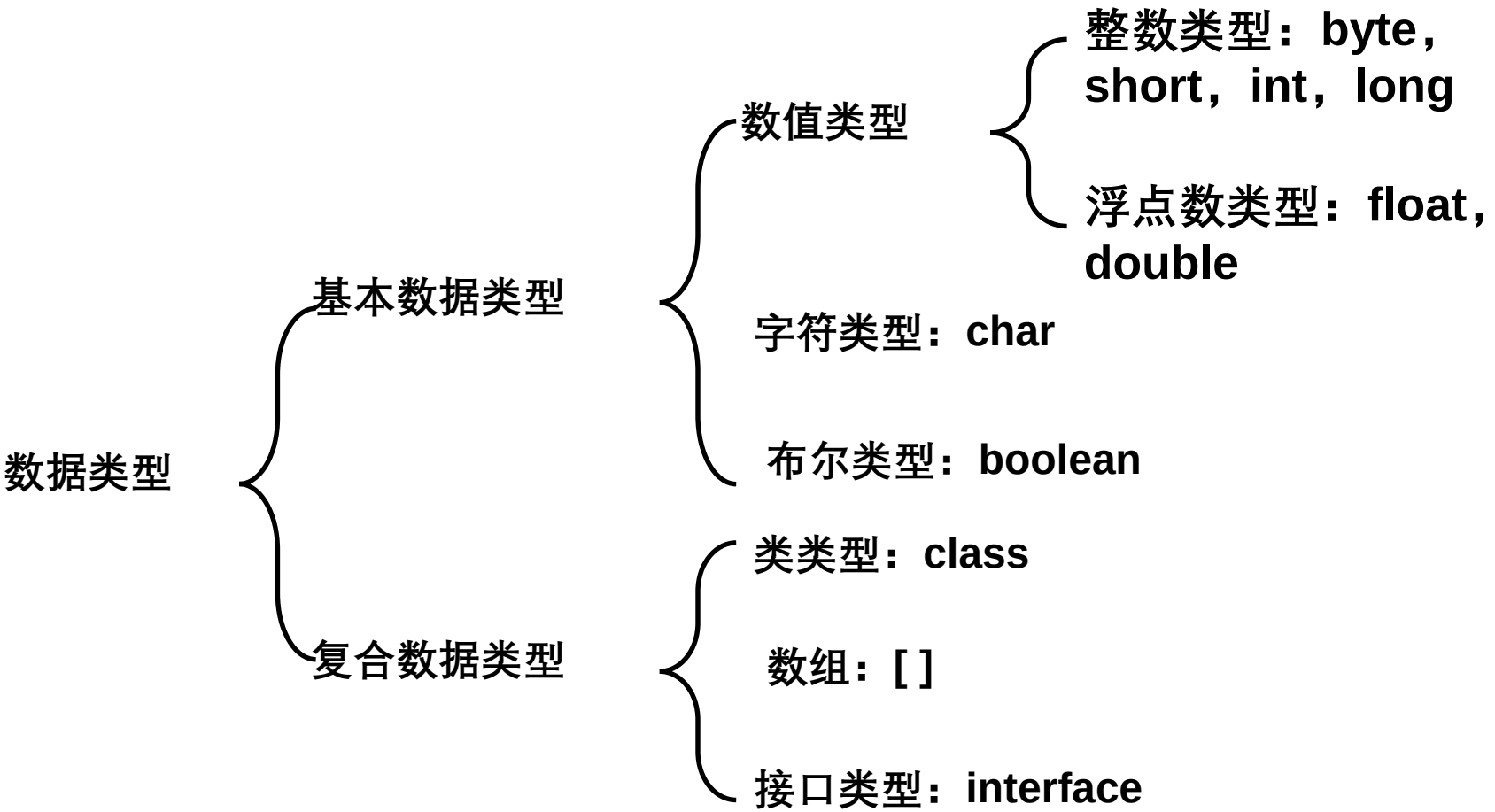
三、程序补充题

一、主要内容

1. 标识符

- Java中标识符的组成：
  - 字母，数字，_，$ 这四种字符
- Java中标识符的开头字符可以是：
  - 字母，_ ，$（数字不能作为开头字符）
- Java中标识符需要注意的地方：
  - 关键字不能作为标识符
  - 标识符区别大小写
  - 数字不能开头
  - 标识符区分大小写，长度没有限制

2、基本数据类型的分类


数据类型
基本数据类型
数值类型
整数类型：byte，short，int，long
浮点数类型：float，double
字符类型：char
布尔类型：boolean
复合数据类型
类类型：class
数组：[ ]
接口类型：interface

1. 数据类型特征表

类型	位数	最小值	最大值	默认值	其他
byte	8	-128(-2^7)	127(2^7-1)	0	有符号、二进制补码表示
short	16	-32768(-2^15)	32767(2^15-1)	0	有符号、二进制补码表示
int	32	-2^31	2^31-1	0	有符号、二进制补码表示
long	64	-2^63	2^63-1	0L(0l)	有符号、二进制补码表示
float	32	2^(-149)	2^128-1	0.0f	单精度、IEEE754标准
double	64	2^(-1074)	2^1024-1	0.0d	双精度、IEEE754标准
char	16	\u0000(0)	\uffff(65535)	\u0000(0)	单一的、Unicode字符

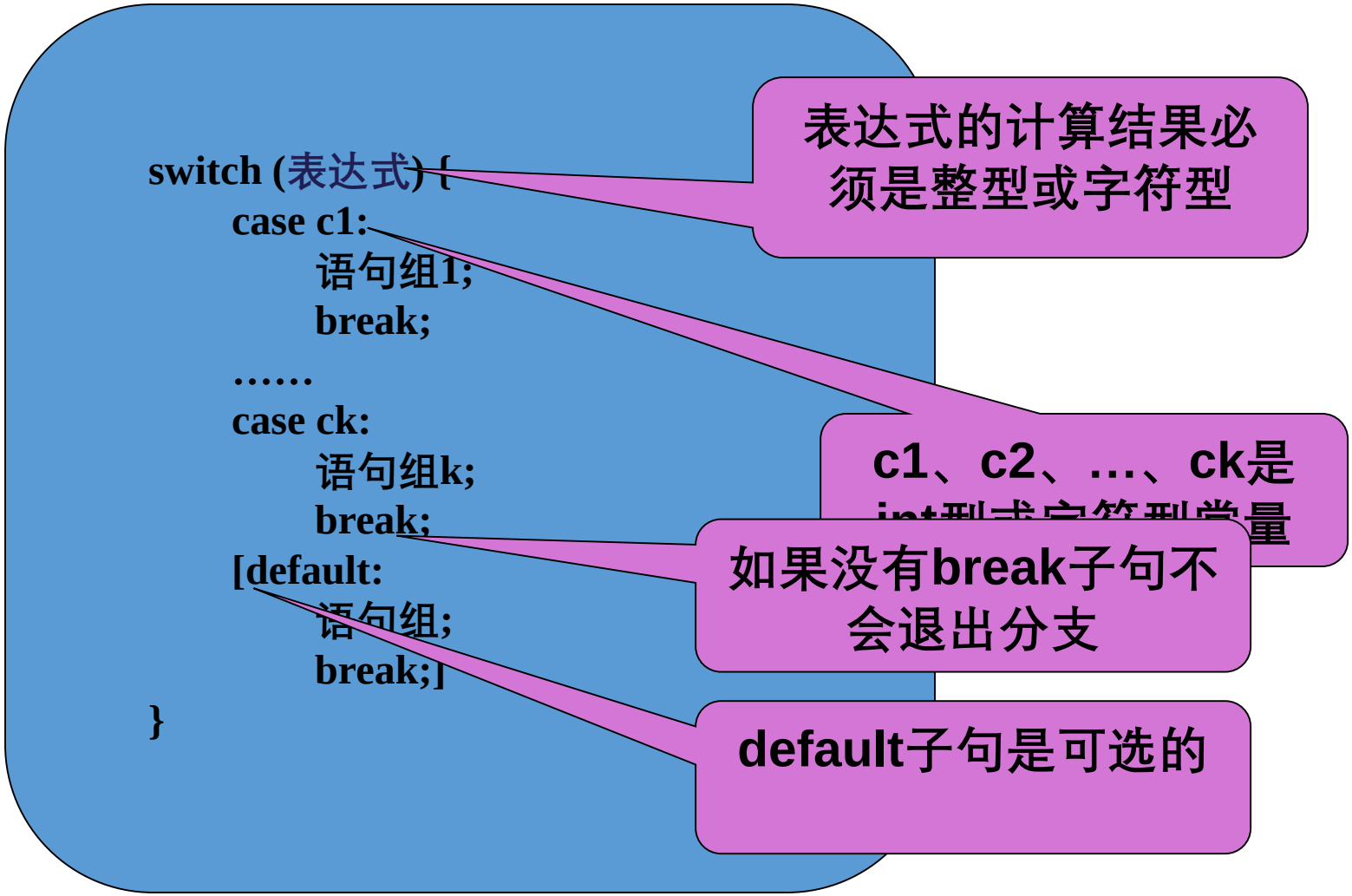
运算符

- 算术运算符
  - 除（/）、取模（%）
  - ++, --
- 关系运算符：> < == <= >= !=
- ! && ||
- 累计赋值“+，－，*，/，%”与“=”结合，如+=、-=、/=、*=

分支语句

- If(逻辑表达式)-else

- switch语句格式


switch (表达式) {
case c1:
语句组1;
break;
……
case ck:
语句组k;
break;
[default:
语句组;
break;]
}
表达式的计算结果必须是整型或字符型
c1、c2、…、ck是
如果没有break子句不会退出分支
default子句是可选的

循环语句

- for语句语法格式：

```
for (初始语句; 逻辑表达式; 迭代语句)
    循环体语句;
```

- while循环的语法是：

```
while (逻辑表达式)
    循环体语句;
```

- do循环的语法是：

```
do
    语句;
while (逻辑表达式);
```

break与continue

- 中断循环
- 中断本次循环，继续下次循环

创建一维数组

类型　数组名=new　类型[数组长度];

注意：类型一致，设置数组长度的位置

成员变量length

通过数组下标表示数组里的各个元素。数组下标范围0~length-1

可以在声明数组的同时，为数组赋初值，这时就不用显式地使用new运算了，如：

```
int  a[]={1，2，3，4，5}；//数组a的5个元素分别得到值：
```

字符串

一种是创建以后不能改变的字符串常量，String类用于存储和处理字符串常量；

一种字符串是创建以后，可对其进行各种修改操作的字符串变量，StringBuffer类用于存储和操作字符串变量。

String类中常用的方法

length():

返回字符串中的字符个数

charAt(int index):

返回字符串中index位置的字符。

toLowerCase():

将当前字符串中所有字符转换为小写形式。

toUpperCase():

将当前字符串中所有字符转换为大写形式。

substring(int beginIndex,int endIndex):

截取beginIndex到endIndex(不包括)的字符串。

static String valueOf(float f):

用单精度浮点数构造字符串对象。

连接

- String
  - 可以使用***concat***(String str)方法将str连接在当前字符串的尾部。
    - 例：s.concat(“a”);
  - 只要“+”左侧的操作数是字符串，则右侧的操作数也自动变为字符串类型。
    - 例：1+2+3+“No”与“No”+1+2+3
- StringBuffer
  - Append()

14.String字符串相等

equals

==

访问控制修饰符

	同一个类	同一个包	不同包的子类	所有类
private	*			
默认	*	*		
protected	*	*	*	
public	*	*	*	*

16.单重继承

- **Java只支持单重继承**，即每个类只能有一个父类。
- 为了保留多重继承的功能，提出了**接口**的概念

构造方法

- 是一种特殊的方法
- 构造方法的名字与类名相同
- 没有返回值类型

- 构造方法的调用:参数匹配
- 如果程序员定义了一个或多个构造方法，则将自动屏蔽掉缺省的构造方法。

static

- static 在变量或方法之前，表明它们是属于类的，称为类方法（静态方法）或类变量（静态变量)。

- 静态方法的调用

```
class  AA {
      static  void  Show( ){ System.out.println(“我喜欢Java!”); }
}
```

final

- final在类之前
  - 表示该类是最终类，不能再被继承。
- final在方法之前
  - 表示该方法是最终方法，该方法不能被任何派生的子类覆盖。
- final在变量之前
  - 表示变量的值在初始化之后就不能再改变；相当于定义了一个常量。

abstract

- 在一个类中声明为抽象方法时，意味着这个方法必须在子类中被重新定义（覆盖）
  - 构造方法不能为抽象的，抽象与最终不能同时存在；
  - 抽象方法一般是仅有方法头而没有方法体的方法；
  - 抽象方法为该类的子类定义一个方法的接口标准，子类必须根据需要重新定义它
- 抽象类必须被继承
  - 任何含有抽象方法的类必须声明为抽象类；

接口

- 接口主要作用是实现多重继承的功能
- 接口的声明与实现

```
[public] interface 接口名 extends [父接口名列表] {
        变量声明;       //常量
        方法声明;       //抽象方法
    }
```

```
[修饰符] class 类名 extends 父类名 implements 接口名1，
                                              [接口名2，...]
```

抽象类和接口的联系

I/O流的概念

- 理解流，标准输入/输出流的概念

- Java流的分类
  - 字节流

    8 位

    InputStream　OutputStream
  - 字符流

    16 位 Unicode

    Reader　Writer

文件的读写

- 文本文件的读写
  - 用FileInputStream读文本文件
  - 用FileOutputStream写文本文件
  - 用DataOutputStream写二进制文件
  - 用DataInputStream读二进制文件

用DataOutputStream 写二进制文件

- **构造一个数据输出流对象**

  FileOutputStream outFile = new FileOutputStream("temp.class");

  DataOutputStream out = new DataOutputStream(outFile);

- **利用文件输出流类的方法写二进制文件**

  out.writeByte(1); //把数据写入二进制文件

- **数据输出流的关闭**

  out.close();

-

系统的输入与输出

- System.out == 系统的监视器(显示器)
- System.in == 系统的输入(键盘)

具体使用（用于封装字符流）：

```
Scanner Sc=new Scanner(System.in);
```

- nextDouble()
- nextInt()
- nextLong()
- nextShort()
- 等等

图形用户界面编程

- 如果希望使用AWT提供的类库，则需要
  import java.awt.*;

- 如果希望使用Swing类库，则需要
  import javax.swing.*;

- 如果希望使用awt的事件处理，则需要
  import java.awt.event.*;

如何处理事件

Java中进行事件处理的一般方法归纳如下：

1. 对于某种类型的事件XXXEvent，要想接收并处理这类事件，必须定义相应的事件监听器类，该类需要实现与该事件相对应的接口XXXListener；

2. 事件源实例化以后，必须进行授权，注册该类事件的监听器，使用addXXXListener(XXXListener) 方法来注册监听器

3. 明确接口所包括的方法，在监听者类中必须实现接口中的所有方法。

布局管理器

- 容器的缺省布局管理器
  - FlowLayout：Jpanel
  - BorderLayout：JFrame JApplet内容面板(content pane)的默认布局管理器、Window、Dialog和Frame