1. 一个".java"源文件中是否可以包括多个类（不是内部类）？有什么限制?

答：可以包括 多个类，但是最多只能有一个public修饰的类

1. switch语句能否作用在byte上，能否作用在long上，能否作用在String上?

答：switch语句是使用int 型和枚举型的，而精度比int 型低的byte char short 都可以转化成int 型，所以byte是可以用在switch上的，但是系统不会将long型的转化成int型，所以不能使用long.而String是个类，而不是基本数据类型，系统也不会自动将String 转化成int的。

1. short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错?

答：前一个错了，s1=s1+1处s1+1会使short型s1转化成int型，但是在将int型的值赋给short型的变量将会报错。后一个是对的，因为+=是一个复合型运算符，复合赋值运算符会自动地将运算结果转型为其左操作数的类型，不会对s1的类型进行改变。

1. String是最基本的数据类型吗?

答：不是。最基本的数据类型是byte char short int long float double boolean这八种。而String是一个final类。

5、String s = "Hello";s = s + " world!";这两行代码执行后，原始的String对象中的内容到底变了没有？

答：没有。原来的对象中的内容还是“Hello”，只是后来的s引用指向的是另一个对象“Hello world”而已，改变的是内存指向的地址

6、 String str1 = "aaa";

String str2 = new String(“aaa”); （已经存在aaa了，所有只在堆内存中创建，再指向字符串）

String str3=”bbb”;

String str4 = "aaabbb";

String str5 = "aaa" + "bbb";

String str6 = str1+”bbb”;

String str7 = str2+”ddd”+”eee”;

请问str1,str2,str5,str6,str7 各创建了几个对象？

答：Str1创建了1个，Str2创建了1个，Str5创建1个，Str6创建2个 它会先创建”bbb”再相加成一个新的对象，Str7创建3个。

7、能否在switch语句中使用String作为case条件?

答：在java7之后可以。

8、 private static String a;

private static String b;

public static void main(String[] args) {

String c = a+b;

System.out.println(null); //第一行

System.out.println(c); //第二行

}

请问2行打印各打印的是什么？

答：第一行：报错

第二行：nullnull

9、String是最基本的数据类型吗?

答：不是，引用类型

10、找出以下代码错误

abstract class Name {  
private String name;  
 public abstract boolean isStupidName(String name) {}  
}  
 答：抽象类里的抽象方法不能有方法体  
11.以下代码有错吗？  
public class Something {  
 void doSomething () {  
 private String s = "";  
 int l = s.length();  
 }  
}

答：局部变量前不能放置任何访问修饰符 (private ， public ，和 protected) 。 final 可以用来修饰局部变量 (final 如同 abstract 和 strictfp ，都是非访问修饰符， strictfp 只能修饰 class 和 method 而非 variable) 。  
12.以下代码有错吗？  
abstract class Something {  
 private abstract String doSomething ();  
}  
答：abstract 的 methods 不能以 private 修饰。 abstract 的 methods 就是让子类 implement( 实现 ) 具体细节的，怎么可以用 private 把 abstract method 封锁起来 ( 同理，abstract method 前不能加 final) 。  
13、以下代码有错吗？  
此处， Something 类的文件名叫 OtherThing.java  
class Something {  
 private static void main(String[] something\_to\_do) {   
 System.out.println("Do something ...");  
 }  
}  
答案 : 正确。从来没有人说过Java的Class名字必须和其文件名相同。但public class 的名字必须和文件名相同。  
14.以下代码有错吗？  
interface Playable {  
 void play();  
}  
interface Bounceable {  
 void play();  
}  
interface Rollable extends Playable, Bounceable {  
 Ball ball = new Ball("PingPang");  
}  
class Ball implements Rollable {  
 private String name;  
 public String getName() {  
 return name;  
 }  
 public Ball(String name) {  
 this.name = name;   
 }  
 public void play() {  
 ball = new Ball("Football");   
 System.out.println(ball.getName());  
 }  
}  
答：错。 "interface Rollable extends Playable, Bounceable" 没有问题。 interface 可继承多个 interfaces ，所以这里没错。问题出在 interface Rollable 里的 "Ball ball = new Ball("PingPang");" 。任何在 interface 里声明的 interface variable ( 接口变量，也可称成员变量 ) ，**默认为 public static final 。**也就是说 "Ball ball = new Ball("PingPang");" 实际上是"public static final Ball ball = new Ball("PingPang");" 。在 Ball 类的 Play() 方法中， "ball = new Ball("Football");" 改变了 ball的 reference ，而这里的 ball 来自 Rollable interface ， Rollable interface 里的 ball 是 public static final 的， final 的 object 是不能被改变 reference 的。因此编译器将在 "ball = new Ball("Football");" 这里显示有错。

15、以下选项错误的是：A

A、short s=1；s=s+1; B、short s=1；s+=1；

16、 写一个Singleton出来，用2种方法。  
答：Singleton模式主要作用是保证在Java应用程序中，一个类Class只有一个实例存在。一般Singleton模式通常有几种种形式:

第一种形式: 定义一个类，它的构造函数为private的，它有一个static的private的该类变量，在类初始化时实例话，通过一个public的 getInstance方法获取对它的引用,继而调用其中的方法。  
Public class Singleton {  
 private Singleton(){}  
 //在自己内部定义自己一个实例，是不是很奇怪？  
 //注意这是private 只供内部调用  
 private static Singleton instance = new Singleton();  
 //这里提供了一个供外部访问本class的静态方法，可以直接访问   
 public static Singleton getInstance() {  
 return instance;   
 }  
}  
第二种形式:  
public class Singleton {  
 private static Singleton instance = null;  
 public static synchronized Singleton getInstance() {  
 //这个方法比上面有所改进，不用每次都进行生成对象，只是第一次   
 //使用时生成实例，提高了效率！

synchronized (Singleton.class) {  
 if (instance==null)  
 instance＝new Singleton();  
 return instance;   
 }

}  
}

17、冒泡排序的实现   
答：class Something {  
 private static void main(String[] arg) {

int [] array={1,3,7,9,10,21};

for(int i=0;i<array.length-1;i++){

for(int j=0;j<array.length-i-1;j++){

if(array[j]>array[j+1]){

int emp= array[j];

array[j]=array[j+1];

array[j+1]=emp;

}

}

}

}

}