1、不属于Scala7种数值类型的是?

A、Char

B、Int

C、Float

D、LongLong

Scala 的7种数值类型是：Byte、Short、Int、Long、Float、Double、Char

2、Scala关于变量定义、赋值，错误的是？

A、val a = 3

B、val a:String = 3

C、var b:Int = 3 ; b = 6

D、var b = "Hello World!" ; b = "123"

String只能存放字符串类型

3、下面输出与其他不一致的是？

A、println("Hello World")

B、print("Hello World\n")

C、printf("Hello %s", "World\n")

D、val w = "World" ; println("Hello $w")

改正：val w = "World" ; println(s"Hello ${w}")

4、关于函数def sum(args:Int\*) = {var r = 0 ; for(arg <- args) r += arg ; r} 输出结果不一致的是？

A、sum(1,2,3)

B、sum(6)

C、sum(2,4)

D、sum(1,1,1,2)

显然

5、下列表达式不是过程的是？

A、def foo(num:Int) = {println(num \* 2)}

B、def foo() {println("Hello World")}

C、def foo(num:Int) = {num \* 2}

D、def foo():Unit = {println("Hello World")}

过程没有返回值和等号

6、下列数组定义与其他不一致的是？

A、val a = Array[Int](0, 0)

B、val a = Array(0, 0)

C、val a = new Array[Int](2)

D、val a = Array[Int](1, 1)

7、下列关于数组缓冲val b = ArrayBuffer[Int]()操作结果的注释说明错误的是？

A、b += 1 // b 等于 ArrayBuffer(1)

B、b += (1,2,3) // b 等于 ArrayBuffer(1,2,3)

C、b += Array(1,2) // b 等于 ArrayBuffer(1,2) 追加集合用 ++=

D、b += (1,2,3,4,5);b.trimEnd(3) // b 等于 ArrayBuffer(1,2)

类型错误

8、关于数组的常用算法，下列说法错误的是？

A、Array(1,7,2,9).sum // 等于 19

B、Array(1,7,2,9).sorted // 等于 Array(9,7,2,1)

C、Array("one","two","three").max // 等于 "two"

D、Array("one","two","three").mkString("-") // 等于 "one-two-three"

Sorted默认是升序的；字符串的比较是按照ASCII码大小比较的

9、关于元组 Tuple 说法错误的是？

A、元组的可以包含不同类型的元素

B、元组是不可变的

C、访问元组第一个元素的方式为 pair.\_1

D、元组最多只有2个元素

最多22个元素

与列表一样，元组也是不可变的，但与列表不同的是元组可以包含不同类型的元素。

10、对于函数

def getGoodsPrice(goods:String) = {

val prices = Map(“book” -> 5, “pen” -> 2, “sticker” -> 1)

prices.getOrElse(goods, 0)

}

结果说法错误的是？

A、getGoodsPrice("book") // 等于 5

B、getGoodsPrice("pen") // 等于 2

C、getGoodsPrice("sticker") // 等于 1

D、getGoodsPrice("sock") // 等于 “sock”

0

11、对于元组val t = (1, 3.14, "Fred")说法错误的是？

A、t\_1 等于 1

B、t 的类型为 Tuple3[Int, Double, java.lang.String]

C、val (first, second, \_) = t // second 等于 3.14

D、t.\_0无法访问， 会抛出异常

少“.”

12、对于拉链操作val t1 = Array(1, 3); val t2 = Array(2,4) ; t1.zip(t2)返回结果正确的是？

A、((1,2,3,4)

B、((1,3),(2,4))

C、((1,2),(3,4))

D、(((1,4),(3,4))

13、下面选项中a的取值不一致的是？

A、val a = 3

B、val a:Double = 1 + 2

C、var a = 1; a += 2

D、val b = 1.+(2); val a = b.toInt

3.0

14、指出在Scala中何种情况下赋值语句x=y=1是合法的？

A、var x={}

B、var x=1

C、var x="test"

D、var x=(1,1)

15、编写一个过程countdown(n:Int)，打印从n到0的数字？

A、

def countdown(n:Int){

0 to n foreach print

}

B、

def countdown(n:Int){

(0 until n).reverse foreach print

}

C、

def countdown(n:Int){

(0 to n).reverse foreach print

}

D、

def countdown(n:Int){

(0 to n-1).reverse foreach print

}

//3210 until 不包含 n 的值

16、表达式for(i <- 1 to 3;j <- 1 to 3; if i != j ) print((10 \* i + j) + " ")输出结果正确的是？

A、11 12 13 21 22 23 31 32 33

B、11 13 21 23 31 33

C、12 13 21 23 31 32

D、11 12 21 22 31 32

17、关于函数def fac(n:Int) = { var r = 1 ; for(i <- 1 to n) r = r \* i ; r} fac(5)输出结果正确的是？

A、15

B、120

C、200

D、300

18、关于函数def fac(n:Int, x:String="x", y:Int=0) = { println(x\*(n+y)) } 输出结果不一致的是？

A、fac(2)

B、fac(1, y=1)

C、fac(1, "x")

D、fac(1, "x", 1)

xx

19、关于数组val a = Array(1,2,3)下列说法错误的是？

A、val b = 2 \* a // b 等于 Array(2,4,6)

B、val b = a.map(\_\*2) // b 等于 Array(2,4,6)

C、val b = for(elem <- a) yield 2 \* elem // b 等于 Array(2,4,6)

D、val b = for(elem <- a if elem % 2 == 0) yield 2 \* elem // b 等于 Array(4)

语法错误

20、对于Map("book" -> 5, "pen" -> 2).map(m => m.\_1 -> m.\_2 \* 2)结果说法正确的是？

A、Map("bookbook" -> 5, "penpen" -> 2)

B、Map("bookbook" -> 10, "penpen" -> 4)

C、Map("book" -> 10, "pen" -> 4)

D、Map("book" -> 5, "pen" -> 2 ,"book" -> 5, "pen" -> 2)

21、对于val prices = Map("book" -> 5, "pen" -> 2, "sticker" -> 1)结果说法错误的是？

A、prices("sticker") // 等于 1

B、(prices + ("shoes" -> 30))("shoes")// 等于 30

C、(prices - "pen")("book") // 等于 5

D、prices("sock") // 等于 0

30||异常

22、对于表达式"New York".partition(\_.isUpper)返回结果正确的是？

A、("New", "York")

B、("NY", "er ork")

C、("er ork", "NY")

D、("New York", "NY")

23、对于表达式

val tokens = "one two three four two two three four".split(" ")

val map = new HashMap[String,Int]

for(key <- tokens){

map(key) = map.getOrElse(key,0) + 1

}

说法正确的是？

A、map(“one”) 等于 2

B、map(“two”) 等于 2

C、map(“three”) 等于 3

D、map(“four”) 等于 2

24、编写一个函数，返回Int数组中最大最小的数字？

A、（没加=号）

def maxmin(nums:Array[Int]){

val max = nums.max

val min = nums.min

(max, min)

}

B、

def maxmin(nums:Array[Int]) = {

nums.max -> nums.min

}

C、

def maxmin(nums:Array[Int]) = {

val max = nums.max

val min = nums.min

(max, min)

}

D、

def maxmin(nums:Array[Int]) = {

val max = nums.sorted.head

val min = nums.sorted.last

(max,min)

}反了

25、对于拉链操作

val key = Array(1,2)

val value = Array(“one”,”two”)

val m = key.zip(value).toMap

返回结果正确的是？

A、m(1) 等于 “one”

B、m(2) 等于 “one”

C、m(“one”) 等于 “1”

D、m(“two”) 等于 1

26、scala中，下面的哪个类定义是不正确的？

A.class Counter{def counter = “counter”}

B.class Counter{val counter = “counter”}

C.class Counter{var counter:String}

D.class Counter{def counter () {}}

需要被定义为抽象类

27、scala的类在实现属性时，下面说法中错误的是？

A. var foo： Scala自动合成一个getter 和一个setter

B. val foo： Scala自动合成一个getter 和一个setter

C. 可以单独定义foo方法

D. 可以单独定义foo\_=方法

Val 修饰的变量没有setter 方法

28、定义类Countter并实例化如下：

Class Countter(name:String){

val a = 1

var b = “counter”

}

val counter = new Countter(“computer”)，下列操作正确的是？

A. counter.name = “cpu”

B. counter.a = 2

C. counter.b = “conter”

D. counter.a = counter.b

Val 不可以赋值

29、类和单例对象间的差别是？

A. 单例对象不可以定义方法，而类可以

B. 单例对象不可以带参数，而类可以

C. 单例对象不可以定义私有属性，而类可以

D. 单例对象不可以继承，而类可以

30、关于辅助构造器，以下说法正确的是？

A. 辅助构造器的必须调用主构造器

B. 辅助构造器的可以直接调用超类的主构造器

C. 辅助构造器的参数可以是任意多个

最多22 个

D. 辅助构造器的名称和类名相同

31、关于主构造器，以下说法错误的是？

A. 主构造器在每个类都可以定义多个

一个

B. 主构造器的参数可以直接放在类名后。

C. 主构造器的会执行类定义中的所有语句。

D. 主构造器中可以使用默认参数。

32、定义类 Class Person(private val name: String){},以下说法正确是？

A. name是对象私有字段

B. name是类私有字段，有私有的getter方法

C. name是类公有字段，有公有的getter和setter方法

D. name是类私有字段，可以在类内部被改变

33、scala 中，以下说法错误是？

A. scala 中， 可以在类中定义类。

B. scala 中， 可以在函数中定义函数。

C. scala 中， 不可以在类中定义object。

D. scala 中， 可以在函数中定义类。

函数或者方法中可以定义类

34、scala 中，类成员的缺省访问级别是？

A. public

B. private

C. protected

D. 以上都不对

35、以下单例对象，定义错误的是？

A. object A{var str = ““}

B. object A(str:String){}

C. object A{def str = ““}

D. object A{val str = ““}

Object不能带参数

36、Scala里，final修饰符的描述正确的是？

A. 类的声明上添加final修饰符确保成员不被子类重载

B. 类的声明上添加final修饰符把整个类声明为final，这个类不能被继承

C. 类的声明上添加final修饰符确保成员只能被子类重载一次

D. 以上说法都是错误的

37、scala语言中，关于List的定义。不正确的是？

A. val list = List(1,2,3)

B. val list = List[Int](1,2,3)

C. val list = List[String](‘a’,’b’,’c’)

D. val list = List[String]()

38、定义一个类，定义正确的是？

A.

class House(name){

def getname = “beijing”

}

B.

class house {

name = “nanjing”

}

C.

class House {

public var counter:String = “shanghai”

}

D.

class house (var name:String){

private def getName = name

}

39、关于辅助构造器，以下说法不正确的是？

A. 辅助构造器的名称为this。

B. 辅助构造器的定义要调用已定义的辅助构造器或主构造器。

C. 辅助构造器可以有任意多个

D. 辅助构造器必须要带参数

40、Scala中的类，以下说法正确的是？

A. Scala中，类内部的不是从构造器定义的代码，都会在主构造器中执行

B. Scala中，类内部的不是字段的部分或者方法定义的代码，都会在主构造器中执行

C. Scala中，类内部的所有的代码，都会在主构造器中执行

D. 以上说法都是错误的。

41、scala 中，类和它的伴生对象说法错误的是？

A. 类和它的伴生对象定义在同一个文件中。

B. 类和它的伴生对象可以有不同的名称。

C. 类和它的伴生对象可以互相访问私有特性。

D. 类和它的伴生对象可以实现既有实例方法又有静态方法。

42、关于对象的apply方法，正确的是？

A. apply方法返回的是伴生类的对象。

B. apply方法不可以带参数。

C. apply方法返回的是Unit。

D. apply方法定义在类里面。对象

43、对于下述代码说法正确的是？

class Cat extends Animal{}

A. Cat是Animal的子类。

B. Animal是Cat的子类

C. Cat是Animal的超类

D. Animal一定是抽象类

44、scala中的类，下面说法正确的是？

A. scala 中，使用extends进行类的扩展。

B. scala 中, 声明为final的类可以被继承。

C. scala中，超类必须是抽象类。

D. scala中，抽象类可以被实例化。

45、scala中，下列描述错误的是？

A. Scala里，字段和方法属于相同的命名空间。

B. Scala一共有四个命名空间。

2

C. Scala里，字段可以重载无参数方法。

D. Scala类和特质属于相同的命名空间。

46、下面定义的代码和哪个选项是等效的？

class ArrayElement(x123: Array[String]) extends Element{

val contents: Array[String] = x123

}

A.

class ArrayElement(var x123: Array[String])extends Element{

val contents: Array[String] = x123

}

B.

class ArrayElement(val x123: Array[String] )extends Element{

val contents: Array[String] = x123

}

C.

class ArrayElement(var contents: Array[String]) extends Element

D.

class ArrayElement(val contents: Array[String] ) extends Element

47、类定义如下，描述不正确的是？

abstract class Element

{

def contents: Array[String]

}

A. Element是一个抽象类。

B. Element不能实例化。

C. Contents成员定义错误，没有初始化。

D. contents方法是类Element的抽象成员。

48、scala语言中，下面描述正确的是？

A. scala中，Float是Double的子类。

B. scala中，Int是Long的子类。

C. scala中，Double是AnyRef的子类

D. scala中，Long是AnyVal的子类

49、Scala中，下面的代码执行正确的是?

A. val list = 1 :: 2 :: 3

B. val list = 1.::(2).:: (3).::(Nil)

C. val list = 1 :: "s" :: "b" :: Nil

D. var list = 1 ::: 2 ::: 3 ::: Nil

**:: 不能用在Int类型集合中**

50、类ArrayElement定义如下,它的子类定义正确的是？

class ArrayElement(val str:String){

def height =1

}

A.

class LineElement(str: String) extends ArrayElement{

def width = str.length

}

B.

class LineElement(s: String) extends ArrayElement(s) {

def width = str.length

}

C.

class LineElement(str: String) extends ArrayElement(str){

def height =1

}

D.

class LineElement(s String) extends ArrayElement{

def str = s

}

51、以下关于闭包描述错误的是？

A. 闭包是一个函数，其返回值依赖于声明在函数外部的一个或多个变量。

B. 通常来讲，可以将闭包看作是可以访问一个函数里面局部变量的另一个函数。

C. 对于def mulBy(factor: Double) = (x: Double) => factor \* x; val triple = mulBy(3);,函数triple是一个闭包。

D. 对于def mulBy(factor: Double) = (x: Double) => 3 \* x; val triple = mulBy(3);,函数triple是一个闭包。

52、有关柯里化描述错误的是？

A. 柯里化是指将原来接受两个参数的函数变成新的接受一个参数的函数的过程。新的函数返回一个以原有第二个参数作为参数的函数。

B. 有时，使用柯里化将某个函数参数单拎出来，可以提供更多用于类型推断的信息。

C. 将函数def add(x: Int, y: Int) = x + y，变形为def add(x: Int)(y: Int) = x + y的过程是一个柯里化过程。

D. 柯里化是多参数列表函数的特例。

53、高阶函数是指？

A. 在程序中应该首先被定义的函数。

B. 将函数作为参数，并返回结果为函数的函数。

C. 函数参数为函数或返回结果为函数的函数。

D. 执行时间长的函数。

54、以下选项中，哪一个可以在Scala中定义一个按名调用(call-by-name)的函数？

A. def functionOne(x: Int)

B. def functionOne(x: => Int)

C. def functionOne(x: -> Int)

D. def functionOne(x: Name)

55、如下代码执行结果是？

def someCurring(y: Int):(Int) => Int = {

def s(y:Int) = 2\*y

s

}

val result = someCurring(4)(3)

A. 6

B. 4

C. 8

D. 9

56、集合可以分为三大类，以下描述中不是这三大类的是？

A. 列表(List)

属于Seq

B. 序列(Seq)

C. 集(Set)

D. 映射(Map)

57、以下对集合的描述有误的是？

A. Set是一组没有先后次序的值。

B. Map是一组(键，值)对偶。

C. 每个Scala集合特质或类都有一个带有apply方法的伴生对象，可以用此方法来构建该集合中的实例。

D. 为了顾及安全性问题，Scala仅支持不可变集合而不支持可变集合。

58、 如下的操作中，哪一个不能将val names = List("Petter", "Paul", "Mary")中的相应地字符全部变成大写？

A. names.map(\_.toUpperCase)

B. for (name <- names) yield name.toUpperCase

C. for (name <- names) yield for(c <- name) yield c.toUpper

D. for(name <- names; c <- name) yield c.toUpper

List[Char] = List(P, E, T, T, E, R, P, A, U, L, M, A, R, Y)

59、以下关于case class描述有误的是？

A. 默认情况下可对其属性进行修改。

B. 在模式匹配时会进行解构操作。

C. 两个case class进行比较时进行全等比较，而非按是否引用同一个对象进行比较。

D. 相比于class而言更简洁的实例化过程(不需要使用new操作符)及其它操作语法。

60、下面有Option类型描述有误的是？

A. Option类型可以用来表示那种可能存在、也可能不存在的值。

B. Option类型的两种状态分别是Some和None。

C. Option类型的两种状态分别是Success和Failure。

D.可以使用isEmpty方法来判断是否存在值。

61、对于以下代码描述有误的是？

val data = Map(1 -> "One", 2 -> "Two")

val res = for((k, v) <- data; if(k > 1)) yield v

A. 运行后res的结果为List("Two")。

B. 运行后res的结果为List("One", "Two")。

C. 对映射data中的每一个(键，值)对，k被绑定对键，而v则被绑定到值。

D. 其中的if(k > 1)是一个守卫表达式。

62、以下关于特质的说法错误的是？

A. 类可以实现任意数量的特质。

B. 特质可以要求实现它们的类具备特定的字段、方法或超类。

C. 与Java接口(Interface)相同，Scala特质也可以提供方法和字段的实现。

D. 当将多个特质叠加在一起时，顺序很重要——其方法先被执行的特质排在更后面。

63、关于Scala中的trait和Java中的Interface描述错误的是？

A. Scala特质和Java接口都可以包含抽象方法和具体实现。

B. Scala和Java都不允许多重继承，但可以叠加多个特质或接口。

C. Java只能在类层面上添加接口的实现，而Scala可以在类和对象层面上”混入”特质。

D. 以上描述至少有一项有误。

64、对于Scala中混入了特质的对象，关于其在构造时构造器执行顺序的描述错误的是？

A. 首先调用超类的构造器。

B. 特质构造器在超类构造器之后、类构造器之前执行。所有特质构造完毕，子类被构造。

C. 特质由右到左被构造。如果多个特质共有一个父特质，而那个父特质已经被构造，则不会再次构造。

D. 每个特质中，父特质先被构造。

65、对集(Set)进行操作"Set(2, 0, 1) + 1 + 1 - 1"之后的结果为？

A. Set(2, 0, 1, 1)

B. Set(2, 0, 1)

C. Set(2, 0)

D. 已上均不正确。

66、下面有关val f: PartialFunction[Char, Int] = {case '+' => 1; case '-' => -1}的描述有误的是？

A. f是一个偏函数，且其参数类型为Char、返回类型为Int。

B. f('-')的结果是返回-1。

C. f.isDefinedAt('0')的结果是返回false。

D. f('0')在执行时由于没有对应的case语句，故会返回一个任意Int值。

异常

67、var，val，lazy val和def三个关键字之间的区别？

1）val是java的final不可变变量，var是java的普通变量；

2）在main函数里，val和var仅声明变量；

3）在class类定义里，val和var是先声明field存储空间，然后分别为他们定了同名的方法。

4）val/var和def的区别，可以从callByValue 和 callByName的角度来理解，val,var是callByValue， def是callByName.

lazy val 用到时候被实例化

68、trait（特质）和abstract class（抽象类）的区别?

优先使用特质. 一个类扩展多个特质是很方便的, 但却只能扩展一个抽象类.   
如果你需要构造函数参数, 使用抽象类. 因为抽象类可以定义带参数的构造函数, 而特质不行.

特质可以被多继承

抽象类只能单继承

69、call-by-value和call-by-name求值策略的区别？

（1）call-by-value是在调用函数之前计算；  
（2）call-by-name是在需要时计算

1. Option类型的定义和使用场景？

在Java中，null是一个关键字，不是一个对象，当开发者希望返回一个空对象时，却返回了一个关键字，为了解决这个问题，Scala建议开发者返回值是空值时，使用Option类型，在Scala中null是Null的唯一对象，会引起异常，Option则可以避免。Option有两个子类型，Some和None（空值）

1. object和class的区别？

object是类的单例对象，不能带参数，开发人员无需用new关键字实例化。如果对象的名称和类名相同，这个对象就是伴生对象

72、 case class 的特点？  
当你声明了一个 case class，Scala 编译器为你做了这些：

创建 case class 和它的伴生 object

实现了 apply 方法让你不需要通过 new 来创建类实例

默认为主构造函数参数列表的所有参数前加 val

添加天然的 hashCode、equals 和 toString 方法。由于 == 在 Scala 中总是代表 equals，所以 case class 实例总是可比较的

生成一个 copy 方法以支持从实例 a 生成另一个实例 b，实例 b 可以指定构造函数参数与 a 一致或不一致

由于编译器实现了 unapply 方法，一个 case class 支持模式匹配

73、伴生对象是什么？  
单例对象与类同名时，这个单例对象被称为这个类的伴生对象，而这个类被称为这个单例对象的伴生类。伴生类和伴生对象要在同一个源文件中定义，伴生对象和伴生类可以互相访问其私有成员。不与伴生类同名的单例对象称为孤立对象。

74、 Scala类型系统中Nil, Null, None, Nothing四个类型的区别  
 None 是一个object，是Option的子类型scala推荐在可能返回空的方法使用Option[X]作为返回类型。如果有值就返回Some[x](Some也是Option的子类)，否则返回None

Null 是所有AnyRef的子类，在scala的类型系统中，AnyRef是Any的子类，同时Any子类的还有AnyVal。对应java值类型的所有类型都是AnyVal的子类。所以Null可以赋值给所有的引用类型(AnyRef)，不能赋值给值类型，这个java的语义是相同的。 null是Null的唯一对象。

Nothing 是所有类型的子类，也是Null的子类。Nothing没有对象，但是可以用来定义类型。例如，如果一个方法抛出异常，则异常的返回值类型就是Nothing(虽然不会返回)

Nil 是一个空的List，定义为List[Nothing]，根据List的定义List[+A]，所有Nil是所有List[T]的子类。

75、隐式参数的优先权

传入>上下文类型一致的隐式值>默认值