16.下列语句：int *p, a = 10; p = &a.

均表示地址的是（）

A.a , p, &a

B.&*a,&a,*p

C.*&p, *p, &a

D.&a, p, &*p

> 解析： 答案：D
>
> - *p表示指针p
> - &a表示取a的内存地址
> - p = &a 表示p等于a的内存地址
> - &*p表示获取指针p的内存地址
> - *&p表示指向P内存地址的一个指针

17.ArrayList list = new ArrayList(20);中的list扩充几次

A.0

B.1

C.2

D.3

> 解析： 答案：A Arraylist默认数组大小是10，扩容后的大小是扩容前的1.5倍，最大值小于Integer 的最大值减8，如果新创建的集合有带初始值，默认就是传入的大小，也就不会扩容，由于本地在创建数组时直接分配了内存大小，所以不会扩充

18.Thread. sleep()是否会抛出checked exception?

A.会

B.不会

> 解析： 答案：A checked exception：指的是编译时异常，该类异常需要本函数必须处理的，用try和catch处理，或者用throws抛出异常，然后交给调用者去处理异常。 runtime exception：指的是运行时异常，该类异常不必须本函数必须处理，当然也可以处理。 Thread.sleep()抛出的InterruptException属于checked exception；IllegalArgumentException属于Runtime exception;

19.关于String、StringBuffer、StringBuilder以下说法错误的是

A.StringBuilder运行速度最快

B.StringBuffer是线程安全的

C.String的值是可变的

D.StringBuffer运行速度比String快

> 解析： 答案：C

- **String**

String不可变！先来看看String的源码：

```
 public final class String implements Serializable, Comparable<String>,
CharSequence {
    private final byte[] value;
    private final byte coder;
    private int hash;
    private static final long serialVersionUID = -6849794470754667710L;
    .......}
```

1、String由final修饰，说明final不能被继承，不能被修改 2、String用来存储数据的是字节数组 byte[]，同样也是由final修饰的。 3、虽然字节数组value是由final修饰，但是我们要清楚一个原则就是：String是引用型变量，要清楚在String的数组是放在堆中的，然后将栈中放的是数组在堆中的引用地址，而我们通常所用的就是一个指向堆中真实数组数据的一个引用地址，所以也称String为引用型对象。简而言之：引用地址不可变，但是地址指向的堆中的数组数据是可以改变的。一旦创建了一个String对象，就在内存中申请一片固定的地址空间存放数据，不管怎么变，地址是不会变的，但是地址所指向的空间真实存放的数据是可以改变的。再通俗点就是：你家的门牌号不会变，但是你家要住几口人是你说了算。

- **StringBuffer**

先来看下StringBuffer的源码：

```
    public synchronized StringBuffer append(String str) {
        this.toStringCache = null;
        super.append(str);
        return this;
    }
```

由于有关键字synchronize，所以线程安全。

> String 和 StringBuffer、StringBuilder 的区别在于 String 声明的是不可变的对象，每次操作都会生成新的 String 对象，然后将指针指向新的 String 对象，而 StringBuffer、StringBuilder 可以在原有对象的基础上进行操作，所以在经常改变字符串内容的情况下最好不要使用 String。

> StringBuffer 和 StringBuilder 最大的区别在于，StringBuffer 是线程安全的，而 StringBuilder 是非线程安全的，但 StringBuilder 的性能却高于 StringBuffer，所以在单线程环境下推荐使用 StringBuilder，多线程环境下推荐使用 StringBuffer。

20.下列关于Java语言中String和char的说法，正确的是（）

A.String是Java定义的一种基本数据类型。

B.String是以"\0"结尾的char类型的数组char[]。

C.使用equals()方法比较两个String是否内容一样（即字符串中的各个字符都一样）。

D.Char类型在Java语言里面存储的是ASCII码。

> 解析： 答案：C
>
> - 基本数据类型包括byte，short，int，long，float，double，char，boolean
> - C语言当中String是以"\0"结尾的char类型的数组char[]，java不是， String内部是用char[]数组实现的，不过结尾不用\0。
> - char存储的unicode码，不仅可以存储ascII码，汉字也可以。

21.如下的Java程序

```
public class Test {
     public static void main(String[] args) {
     System.out.println(args[0]);
     }
}
```

若采用命令行"java Test one two three"调用，则程序输出的结果为：

A.Test

B.one

C.two

D.java

> 解析： 答案：B 采用命令行" java Test one two three "调用 其中Test为调用的方法，而one two three则为Test方法里面main函数的参数； System.out.println(args[0]);表示输出第一个元素，故为one；

22.从内存实现或者反射的角度来看，关于继承的说法正确的是（）。

注：此处的继承不代表能调用

A.子类将继承父类的所有的数据域和方法

B.子类将继承父类的其可见的数据域和方法

C.子类只继承父类public方法和数据域

D.子类只继承父类的方法，而不继承数据域

> 解析： 答案：A 在一个子类被创建的时候，首先会在内存中创建一个父类对象，然后在父类对象外部放上子类独有的属性，两者合起来形成一个子类的对象。所以所谓的继承使子类拥有父类所有的属性和方法其实可以这样理解，子类对象确实拥有父类对象中所有的属性和方法，但是父类对象中的私有属性和方法，子类是无法访问到的，只是拥有，但不能使用。就像有些东西你可能拥有，但是你并不能使用。所以子类对象是绝对大于父类对象的，所谓的子类对象只能继承父类非私有的属性及方法的说法是错误的。可以继承，只是无法访问到而已。

23.java中 String str = "hello world"下列语句错误的是?

A.str+=' a'

B.int strlen = str.length

C.str=100

D.str=str+100

> 解析： 答案：ABC
>
> - str += 'a' 和 str +="a"都是对的，但是如果a前面加一个空格，那么只能用双引号了。代表字符串
> - str += 'a' 和 str +="a"都是对的，但是如果a前面加一个空格，那么只能用双引号了。代表字符串
> - int 无法直接转成String类型

24.下列有关JAVA异常处理的叙述中正确的是（）

A.finally是为确保一段代码不管是否捕获异常都会被执行的一段代码

B.throws是用来声明一个成员方法可能抛出的各种非运行异常情况

C.final用于可以声明属性和方法，分别表示属性的不可变及方法的不可继承

D.throw是用来明确地抛出一个异常情况

> 解析： 答案：ABD

> - throw 用于方法块里面的 代码，比throws的层次要低，比如try...catch ....语句块，表示它抛出异常，但它不会处理它

final

> - 用于类 ---- 说明该类无法被继承，实例：String类
> - 用于方法-----说明该方法无法被覆盖，实例：final不能与abstract关键字同时使用
> - final用于变量-----说明属性不可变（可用于静态和非静态属性），但多和staic连用，表示常量

25.判断对错。List，Set，Map都继承自继承Collection接口。

A.对

B.错

> 解析： 答案：B
>
> 
>

26.以下哪个命令用于查看tar（backup.tar）文件的内容而不提取它？()

A.tar -xvf backup.tar

B.tar -tvf backup.tar

C.tar -svf backup.tar

D.none of these

> 解析： 答案：B 把常用的tar解压命令总结下，当作备忘：
>
> **tar**
>
> -c: 建立压缩档案 -x：解压 -t：查看内容 -r：向压缩归档文件末尾追加文件 -u：更新原压缩包中的文件 这五个是独立的命令，压缩解压都要用到其中一个，可以和别的命令连用但只能用其中一个。下面的参数是根据需要在压缩或解压档案时可选的。 -z：有gzip属性的 -j：有bz2属性的 -Z：有compress属性的 -v：显示所有过程 -O：将文件解开到标准输出 下面的参数-f是必须的 -f: 使用档案名字，切记，这个参数是最后一个参数，后面只能接档案名。
>
> **压缩**
>
> - tar –cvf jpg.tar *.jpg 将目录里所有jpg文件打包成tar.jpg
> - tar –czf jpg.tar.gz *.jpg 将目录里所有jpg文件打包成jpg.tar后，并且将其用gzip压缩，生成一个gzip压缩过的包，命名为jpg.tar.gz
> - tar –cjf jpg.tar.bz2 *.jpg 将目录里所有jpg文件打包成jpg.tar后，并且将其用bzip2压缩，生成一个bzip2压缩过的包，命名为jpg.tar.bz2
> - tar –cZf jpg.tar.Z *.jpg 将目录里所有jpg文件打包成jpg.tar后，并且将其用compress压缩，生成一个umcompress压缩过的包，命名为jpg.tar.Z
> - rar a jpg.rar *.jpg rar格式的压缩，需要先下载rar for linux
> - zip jpg.zip *.jpg zip格式的压缩，需要先下载zip for linux
>
> **解压**
>
> - tar –xvf file.tar 解压 tar包
> - tar -xzvf file.tar.gz 解压tar.gz
> - tar -xjvf file.tar.bz2 解压 tar.bz2
> - tar –xZvf file.tar.Z 解压tar.Z
> - unrar e file.rar 解压rar
> - unzip file.zip 解压zip
>
> **总结**
>
> - *.tar 用 tar –xvf 解压
> - *.gz 用 gzip -d或者gunzip 解压
> - *.tar.gz和*tgz 用 tar –xzf 解压
> - *.bz2 用 bzip2 -d或者用bunzip2 解压
> - *.tar.bz2用tar –xjf 解压
> - *.Z 用 uncompress 解压
> - *.tar.Z 用tar –xZf 解压
> - *.rar 用 unrar e解压
> - *.zip 用 unzip 解压

27.以下哪个类包含方法flush()？ ()

A.InputStream

B.OutputStream

C.A和B 选项都包含

D.A和B 选项都不包含

> **答案:B**
>
> **解析:**
>
> flush（）函数强制将缓冲区中的字符流、字节流等输出，目的是如果输出流输出到缓冲区完成后，缓冲区并没有填满，那么缓冲区将会一直等待被填满。所以在关闭输出流之前要调用flush（）。

28.Java数据库连接库JDBC用到哪种设计模式?

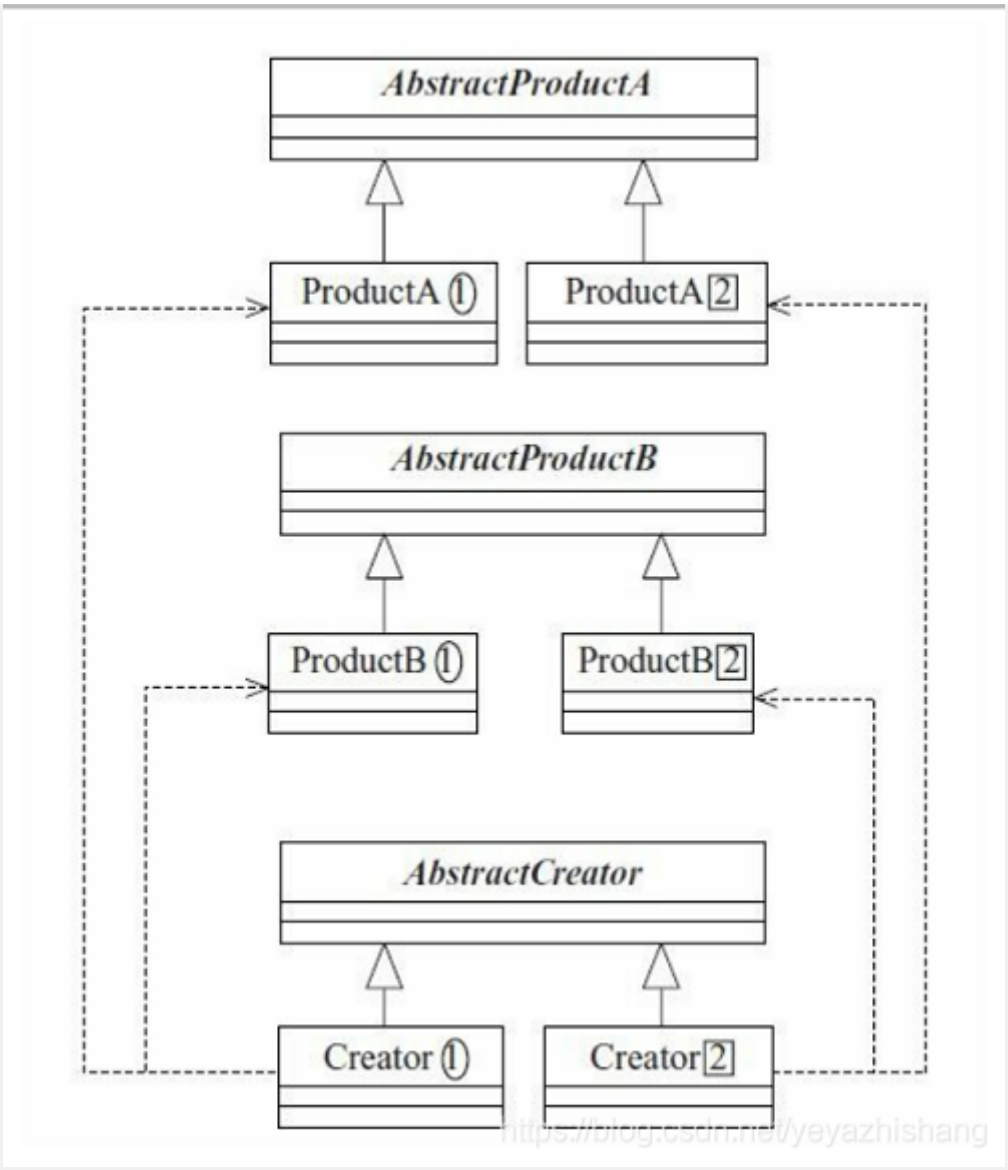
A.生成器

B.桥接模式

C.抽象工厂

D.单例模式

> **答案：B**
>
> **解析:**
>
> - 桥接模式： 定义 ： 将抽象部分与它的实现部分分离，使它们都可以独立地变化。 意图 ： 将抽象与实现解耦。 桥接模式所涉及的角色
>
> 1. Abstraction ： 定义抽象接口，拥有一个Implementor类型的对象引用
> 2. RefinedAbstraction ： 扩展Abstraction中的接口定义
> 3. Implementor ： 是具体实现的接口，Implementor和RefinedAbstraction接口并不一定完全一致，实际上这两个接口可以完全不一样Implementor提供具体操作方法，而Abstraction提供更高层次的调用
> 4. ConcreteImplementor ： 实现Implementor接口，给出具体实现 Jdk中的桥接模式：JDBC JDBC连接 数据库 的时候，在各个数据库之间进行切换，基本不需要动太多的代码，甚至丝毫不动，原因就是JDBC提供了统一接口，每个数据库提供各自的实现，用一个叫做数据库驱动的程序来桥接就行了
>
> - 抽象工厂 抽象工厂模式包含了几个角色： AbstractFactory：用于声明生成抽象产品的方法 ConcreteFactory：实现了抽象工厂声明的生成抽象产品的方法，生成一组具体产品，这些产品构成了一个产品族，每一个产品都位于某个产品等级结构中； AbstractProduct：为每种产品声明接口，在抽象产品中定义了产品的抽象业务方法； Product：定义具体工厂生产的具体产品对象，实现抽象产品接口中定义的业务方法。 这是它的通用类图：



其中AbstractProductA 和 AbstractProductB 就是两个产品族的抽象类 (或者接口)，而 Product1 和 Product2 就是产品族下的具体产品类，AbstractCreator 就是工厂的抽象。

29.以下代码的输出结果是?

```
public class B
{
    public static B t1 = new B();
    public static B t2 = new B();
    {
        System.out.println("构造块");
    }
    static
    {
        System.out.println("静态块");
    }
    public static void main(String[] args)
    {
        B t = new B();
```

```
    }
}
```

A.静态块 构造块 构造块 构造块

B.构造块 静态块 构造块 构造块

C.构造块 构造块 静态块 构造块

D.构造块 构造块 构造块 静态块

答案：C

解析：

大致的执行过程如下：

- 1.实例化t之后，先执行实例化t1，此时会默认执行构造函数，即执行

```
{
        System.out.println("构造块");
    }
```

所以控制台输出构造块，紧接着实例化t2，继续执行构造函数所以控制台接着打印构造块，现在控制台情况如下：

```
D:\soft\java\jdk\bin\java.exe -agentlib:jdwp=transport=dt_socket,address=127.0.0.1
Connected to the target VM, address: '127.0.0.1:53124', transport: 'socket'
构造块
构造块
```

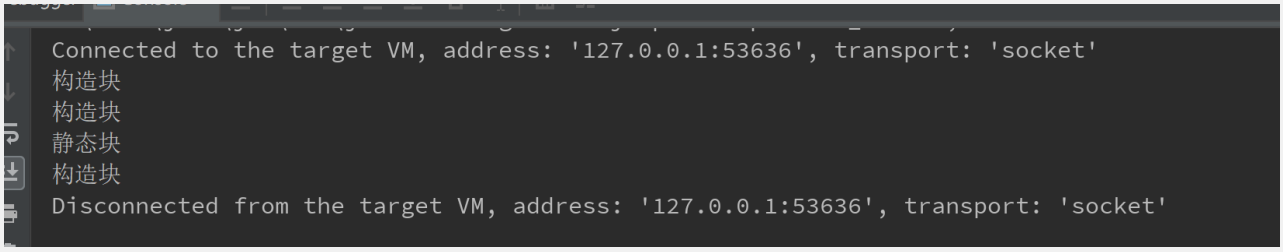
当

```
public static B t1 = new B();
public static B t2 = new B();
```

执行完成之后接着就会按照顺序执行静态代码块

```
static
    {
        System.out.println("静态块");
    }
```

此时输出静态块，最后会执行t的构造函数，即输出构造块



30.下面哪几个函数 public void example(){....} 的重载函数？ ()

A.public void example(int m){...}

B.public int example(){..}

C.public void example2(){..}

D.public int example(int m,float f){...}

答案：A,D

解析：

重载只要求参数列表不同，返回值无关。