31.Java 语言用以下哪个类来把基本类型数据封装为对象（）

A.包装类

B.Class

C.Math

D.Object

答案：A

解析：

> java的数据类型分为两大类：基本类型和引用类型； 基本类型只能保存一些常量数据，引用类型除了可以保存数据，还能提供操作这些数据的功能； 为了操作基本类型的数据，java也对它们进行了封装， 得到八个类，就是java中的基本类型的封装类；他们分别是： 八种基本类型： byte short int long float double char boolean 对应的包装类： Byte Short Integer Long Float Double Character Boolean

32.有以下类定义：

```
abstract class Animal{
    abstract void say();
}
public class Cat extends Animal{
    public Cat(){
        System.out.printf("I am a cat");
    }
    public static void main(String[] args) {
        Cat cat=new Cat();
    }
}
```

运行后：

A.I am a cat

B.Animal能编译，Cat不能编译

C.Animal不能编译，Cat能编译

D.编译能通过，但是没有输出结果

答案：B

解析：

> 包含抽象方法的类称为抽象类，但并不意味着抽象类中只能有抽象方法，它和普通类一样，同样可以拥有成员变量和普通的成员方法。注意，抽象类和普通类的主要有三点区别： 1）抽象方法必须为public或者protected（因为如果为private，则不能被子类继承，子类便无法实现该方法），缺省情况下默认为public。 2）抽象类不能用来创建对象； 3）如果一个类继承于一个抽象类，则子类必须实现父类的抽象方法。如果子类没有实现父类的抽象方法，则必须将子类也定义为为abstract类。 在其他方面，抽象类和普通的类并没有区别。

33.有时为了避免某些未识别的异常抛给更高的上层应用，在某些接口实现中我们通常需要捕获编译运行期所有的异常， catch 下述哪个类的实例才能达到目的：（）

A.Error

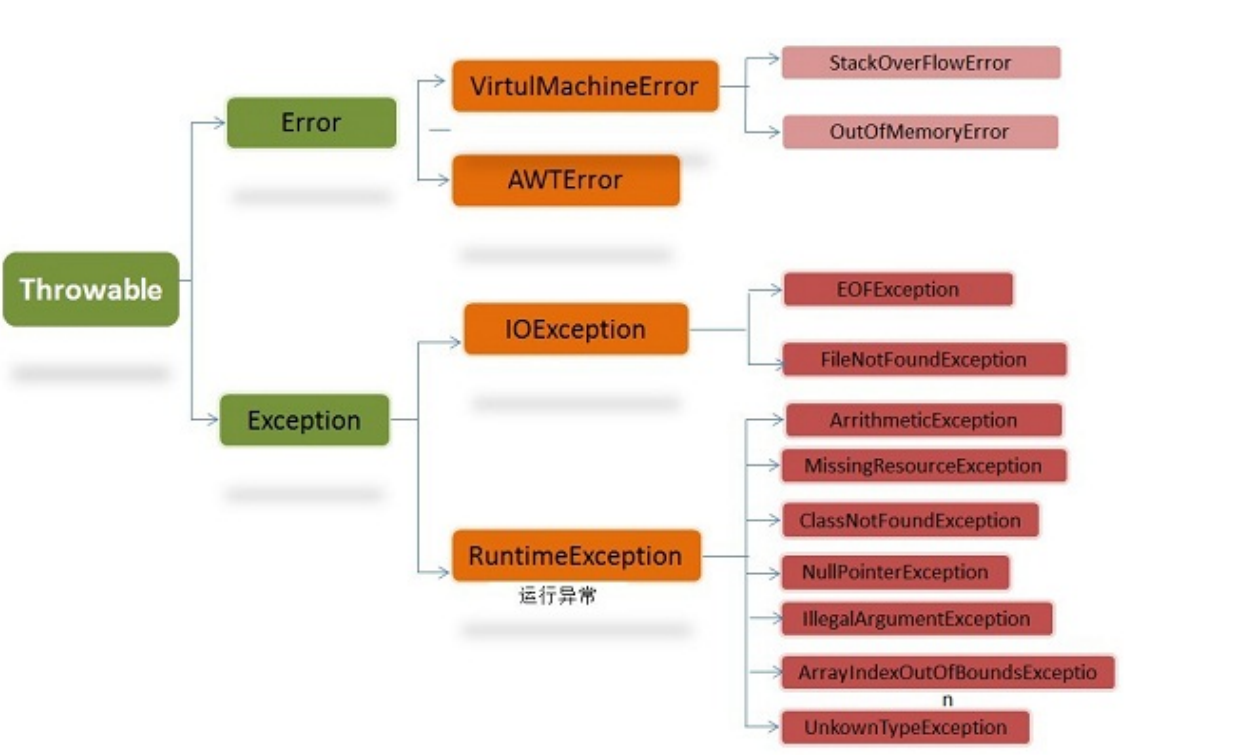
B.Exception

C.RuntimeException

D.Throwable

答案：B

解析：

> 
>
>
> 因为error是系统出错，catch是无法处理的，难以修复的，RuntimeException不需要程序员进行捕获处理，error和exception都是throwable的子类，我们只需要对exception的实例进行捕获即可

34.关于abstract类的声明和描述

> 1、abstract类不能用来创建abstract类的对象； 2、final类不能用来派生子类，因为用final修饰的类不能被继承； 3、如2所述，final不能与abstract同时修饰一个类，abstract类就是被用来继承的； 4、类中有abstract方法必须用abstract修饰，但abstract类中可以没有抽象方法，接口中也可以有abstract方法。

35.下面有关java threadlocal说法正确的有?

A.ThreadLocal存放的值是线程封闭，线程间互斥的，主要用于线程内共享一些数据，避免通过参数来传递

B.线程的角度看，每个线程都保持一个对其线程局部变量副本的隐式引用，只要线程是活动的并且ThreadLocal 实例是可访问的；在线程消失之后，其线程局部实例的所有副本都会被垃圾回收

C.在Thread类中有一个Map，用于存储每一个线程的变量的副本。

D.对于多线程资源共享的问题，同步机制采用了“以时间换空间”的方式，而ThreadLocal采用了“以空间换时间”的方式

答案：B

解析：

> - ThreadLocal类用来提供线程内部的局部变量。这种变量在多线程环境下访问(通过get或set方法访问)时能保证各个线程里的变量相对独立于其他线程内的变量。ThreadLocal实例通常来说都是private static类型的，用于关联线程和线程的上下文。 可以总结为一句话：ThreadLocal的作用是提供线程内的局部变量，这种变量在线程的生命周期内起作用，减少同一个线程内多个函数或者组件之间一些公共变量的传递的复杂度。
>
> 举个例子，我出门需要先坐公交再做地铁，这里的坐公交和坐地铁就好比是同一个线程内的两个函数，我就是一个线程，我要完成这两个函数都需要同一个东西：公交卡（北京公交和地铁都使用公交卡），那么我为了不向这两个函数都传递公交卡这个变量（相当于不是一直带着公交卡上路），我可以这么做：将公交卡事先交给一个机构，当我需要刷卡的时候再向这个机构要公交卡（当然每次拿的都是同一张公交卡）。这样就能达到只要是我(同一个线程)需要公交卡，何时何地都能向这个机构要的目的。 有人要说了：你可以将公交卡设置为全局变量啊，这样不是也能何时何地都能取公交卡吗？但是如果有很多个人（很多个线程）呢？大家可不能都使用同一张公交卡吧(我们假设公交卡是实名认证的)，这样不就乱套了嘛。现在明白了吧？这就是ThreadLocal设计的初衷：提供线程内部的局部变量，在本线程内随时随地可取，隔离其他线程。
> - ThreadLocal类用于创建一个线程本地变量 在Thread中有一个成员变量ThreadLocals，该变量的类型是ThreadLocalMap,也就是一个Map，它的键是threadLocal，值为就是变量的副本。通过ThreadLocal的get()方法可以获取该线程变量的本地副本，在get方法之前要先set,否则就要重写initialValue()方法。 ThreadLocal的使用场景：
>
> 数据库连接：在多线程中，如果使用懒汉式的单例模式创建Connection对象，由于该对象是共享的，那么必须要使用同步方法保证线程安全，这样当一个线程在连接数据库时，那么另外一个线程只能等待。这样就造成性能降低。如果改为哪里要连接数据库就来进行连接，那么就会频繁的对数据库进行连接，性能还是不高。这时使用ThreadLocal就可以既可以保证线程安全又可以让性能不会太低。但是ThreadLocal的缺点时占用了较多的空间。

36.判断一块内存空间是否符合垃圾收集器收集的标准有哪些?

A.给对象赋予了空值null,以下再没有调用过

B.对象重新分配了内存空间

C.给对象赋予了空值null

D.给对象赋予了新值

答案：B

解析：

> 在java语言中，判断一块内存空间是否符合垃圾收集器收集标准的标准只有两个： 1.给对象赋值为null，以下没有调用过。 2.给对象赋了新的值，重新分配了内存空间。

37. 父类，子类的方法以及静态代码块执行的顺序

> 执行顺序为
>
> - 1.父类静态代码块、静态变量 ps:按声明顺序执行
> - 2.子类静态代码块、静态变量 ps:按声明顺序执行
> - 3.父类局部代码块、成员变量 ps:按声明顺序执行
> - 4.父类构造函数
> - 5.子类局部代码块、成员变量 ps:按声明顺序执行
> - 6.子类构造函数

38.

```
Integer a = 1;
Integer b = 1;
Integer c = 500;
Integer d = 500;
System.out.print(a == b);
System.out.print(c == d);
```

A.true、true

B.true、false

C.false、true

D.false、false

答案：B

解析：

> Integer a = 1;是自动装箱会调用Interger.valueOf(int)方法；该方法注释如下： This method will always *** values in the range -128 to > 127 inclusive, and may *** other values outside of this range. 也就是说IntegerCache类缓存了-128到127的Integer实例，在这个区间内调用valueOf不会创建新的实例。

39.访问权限控制从最大权限到最小权限依次为：public、 包访问权限、protected和private 。 （ ）

A.正确

B.错误

答案：B

解析:

表 5.1 访问控制级别表

	private	default	protected	public
同一个类中	√	√	√	√
同一个包中		√	√	√
子类中			√	√
全局范围内				√

40.（）运算符把其操作数中所有值为0和所有值为1的位分别在结果的相应中设置1和0

A.&

B.|

C.!

D.~

答案：D

解析：

> ~是位运算符，意义是 按位非（NOT） 按位非也叫做补，一元运算符NOT"~"是对其运算数的每一位取反。 仅用于整数值 反转位，即0位变为1位，1变成0 在所有情况下∽x等于（-x）- 1

例如：

> ~ 0111 (7) = 1000 (8)

41.在java中重写方法应遵循规则的包括（）

A.访问修饰符的限制一定要大于被重写方法的访问修饰符

B.可以有不同的访问修饰符

C.参数列表必须完全与被重写的方法相同

D.必须具有不同的参数列表

答案：B C

解析：

> 方法的重写（override）两同两小一大原则： 方法名相同，参数类型相同 子类返回类型小于等于父类方法返回类型， 子类抛出异常小于等于父类方法抛出异常， 子类访问权限大于等于父类方法访问权限。

42.Java创建对象的方式

Java有5种方式来创建对象：

- 1.使用 new 关键字（最常用）

```
ObjectName obj = new ObjectName();
```

- 2.使用反射的Class类的newInstance()方法

```
ObjectName obj = ObjectName.class.newInstance();
```

- 3.使用反射的Constructor类的newInstance()方法

```
 ObjectName obj = ObjectName.class.getConstructor.newInstance();
```

- 4.使用对象克隆clone()方法：

```
ObjectName obj = obj.clone();
```

- 5.使用反序列化（ObjectInputStream）的readObject()方法：

```
try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) { ObjectName obj = ois.readObject(); }
```

43.下列哪些操作会使线程释放锁资源?

A.sleep()

B.wait()

C.join()

D.yield()

答案：B C

解析：

- 1.sleep()方法

> 在指定时间内让当前正在执行的线程暂停执行，但不会释放“锁标志”。不推荐使用。 sleep()使当前线程进入阻塞状态，在指定时间内不会执行。

- 2.wait()方法

> 在其他线程调用对象的notify或notifyAll方法前，导致当前线程等待。线程会释放掉它所占有的“锁标志”，从而使别的线程有机会抢占该锁。 当前线程必须拥有当前对象锁。如果当前线程不是此锁的拥有者，会抛出IllegalMonitorStateException异常。 唤醒当前对象锁的等待线程使用notify或notifyAll方法，也必须拥有相同的对象锁，否则也会抛出IllegalMonitorStateException异常。 waite()和notify()必须在synchronized函数或synchronized block中进行调用。如果在non-synchronized函数或non-synchronized block中进行调用，虽然能编译通过，但在运行时会发生IllegalMonitorStateException的异常。

- 3.yield方法

> 暂停当前正在执行的线程对象。 yield()只是使当前线程重新回到可执行状态，所以执行yield()的线程有可能在进入到可执行状态后马上又被执行。 yield()只能使同优先级或更高优先级的线程有执行的机会。

- 4.join方法

> join()等待该线程终止。 等待调用join方法的线程结束，再继续执行。如：t.join();//主要用于等待t线程运行结束，若无此句，main则会执行完毕，导致结果不可预测

44.以下程序运行的结果为（）

```
public class Example extends Thread{
@Override

    public void run(){

        try {

            Thread.sleep(1000);

        } catch (InterruptedException e){

            e.printStackTrace();

        }

        System. out .print( "run" );

    }

    public static void main(String[] args){

        Example example= new Example();

        example.run();

        System. out .print( "main" );

    }
}
```

A.run main

B.main run

C.main

D.run

E.不能确定

答案：A

解析：

> 为啥是runmain而不是mainrun呢？
>
> - 因为启动线程是调用start方法。
> - 把线程的run方法当普通方法，就直接用实例.run()执行就好了。
> - 没有看到start。所以是普通方法调用。

45.下面有关servlet service描述错误的是?

A.不管是post还是get方法提交过来的连接，都会在service中处理

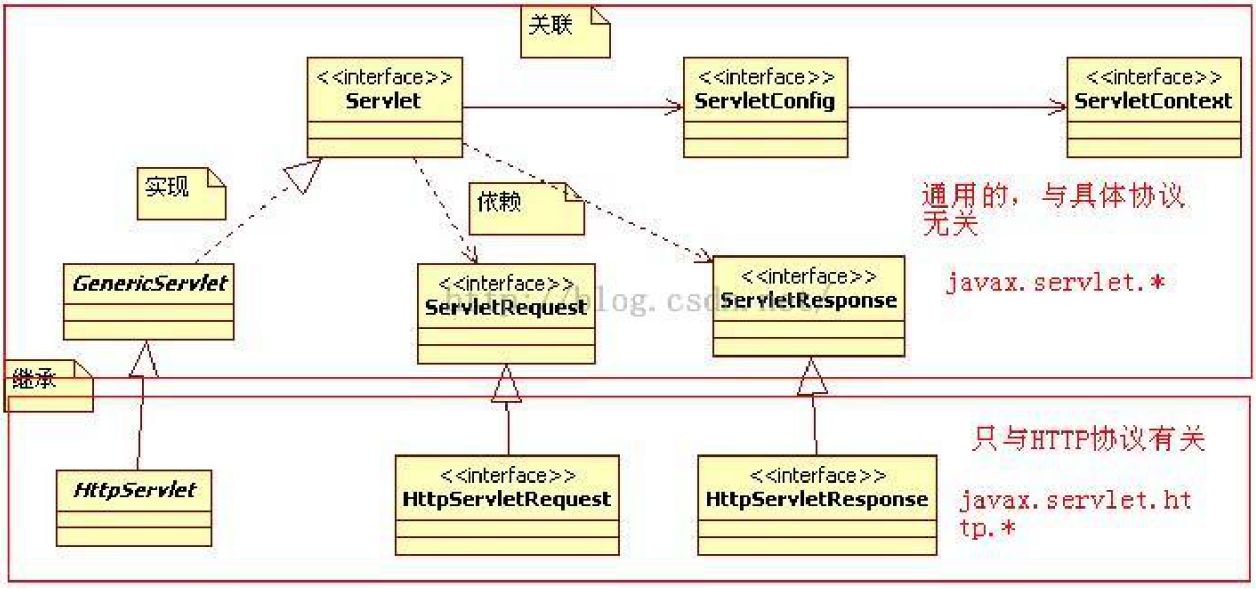
B.doGet/doPost 则是在 javax.servlet.GenericServlet 中实现的

C. service()是在javax.servlet.Servlet接口中定义的

D.service判断请求类型，决定是调用doGet还是doPost方法

答案：B

解析：



doget/dopost与Http协议有关，是在 javax.servlet.http.HttpServlet 中实现的