Java 基础部分

java基本数据类型有哪些，int， long占几个字节

- byte boolean 1位
- char short 2位
- int float 4位
- long double 8位

1.抽象类与接口的区别？

2.分别讲讲 final，static，synchronized 关键字可以修饰什么，以及修饰后的作用？

```
final
final数据
对于基本数据类型的数据而言，final修饰符表示，该数据不会被修改。
对于非基本类型的对象引用而言，final修饰符所限定的引用恒定不变。
final方法–不能被重载
final类–不能被继承，final类中的方法默认为final

```

3.请简述一下String、StringBuffer和StringBuilder的区别？

```
1.String为固定长度的字符串，StringBuilder和StringBuffer为变长字符串；
2.stringBuffer是线程安全的，StringBuilder是非线程安全的；
3.StringBuffer和StringBuilder的默认初始容量是16，可以提前预估好字符串的长度，进一步减少扩容带来的额外开销。
```

4.“equals”与“==”、“hashCode”的区别和使用场景？

```
==是判断两个变量或实例是不是指向同一个内存空间，equals是判断两个变量或实例所指向的内存空间的值是不是相同
==是指对内存地址进行比较 ， equals()是对字符串的内容进行比较
==指引用是否相同， equals()指的是值是否相同
```

5.Java 中深拷贝与浅拷贝的区别？

```
浅拷贝：在拷贝一个对象时，对对象的基本数据类型的成员变量进行拷贝，但对引用类型的成员变量只进行引用的传递，并没有创建一个新的对象，当对引用类型的内容修改会影响被拷贝的对象。
深拷贝：在拷贝一个对象时，除了对基本数据类型的成员变量进行拷贝，对引用类型的成员变量进行拷贝时，创建一个新的对象来保存引用类型的成员变量。
```

6.谈谈Error和Exception的区别？

```
Error：Error属于程序无法处理的错误，是JVM需要负担的责任，无法通过try-catch来进行捕获。
Exception：程序本身可以处理的异常，可以通过catch来进行捕获，通常遇到这种错误，应对其进行处理，使应用程序可以继续正常运行。
```

7.什么是反射机制？反射机制的应用场景有哪些？

```
在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法，对于任意一个对象，都能调用它的任意一个方法和属性；
JDBC 的数据库的连接
Spring 框架的使用
```

8.谈谈如何重写equals()方法？为什么还要重写hashCode()?

```
如何重写

为什么重写hashcode
原因：
1.为了提高效率，hash类型的存储结构，添加元素重复性校验的标准就是先取hashCode值，后判断equals()。重写后，使用hashcode方法提前校验，可以避免每一次比对都调用equals方法
2.为了保证同一个对象，如果重写了equals方法，而没有重写hashcode方法，会出现equals相等的对象，hashcode不相等的情况，重写hashcode方法就是为了避免这种情况的出现
```

9.Java 中 IO 流分为几种?BIO,NIO,AIO 有什么区别?

10.谈谈你对Java泛型中类型擦除的理解，并说说其局限性?

11.String为什么要设计成不可变的?

```
1、字符串常量池
在Java中，由于会大量的使用String常量，如果每一次声明一个String都创建一个String对象，那将会造成极大的空间资源的浪费。
2、使多线程安全
3、避免安全问题
在网络连接和数据库连接中字符串常常作为参数，例如，网络连接地址URL，文件路径path，反射机制所需要的String参数。
4、加快字符串处理速度
由于String是不可变的，保证了hashcode的唯一性，于是在创建对象时其hashcode就可以放心的缓存了，不需要重新计算。所以Map喜欢将String类型的数据作为key
```

12.说说你对Java注解的理解?

13.谈一谈Java成员变量，局部变量和静态变量的创建和回收时机?

14.请说说Java中String.length()的运作原理?

数据结构

1.谈谈List,Set,Map的区别?

```
1. List
有序、可以重复。
ArrayList 底层是数组，查询快，增删慢，线程不安全，效率高，适合get和set方法；
Vector(向量) 底层是数组，查询快，增删慢，线程安全，效率低；
LinkedList 底层是双向循环链表，查询慢，增删快，线程不安全，效率高，适合增加和删除方法。

2. Set
无序、不可重复。
HashSet:（无序，唯一）底层是哈希表，通过 hashCode（）和 equals（）保证元素唯一；
LinkedHashSet：（有序，唯一）底层是链表和哈希表 ，链表保证元素的有序， 哈希表证元素的唯一性；
TreeSet：（有序，唯一）底层是红黑树，排序通过自然排序和比较强排序。

```

2.谈谈ArrayList和LinkedList的区别?

```
1.ArrayList是实现了基于动态数组的数据结构，LinkedList是基于链表结构。
2.对于随机访问的get和set方法，ArrayList要优于LinkedList，因为LinkedList要移动指针。
3.对于新增和删除操作add和remove，LinkedList比较占优势，因为ArrayList要移动数据。
```

3.请说一下HashMap与HashTable的区别

4.谈一谈ArrayList的扩容机制?

5.HashMap 的实现原理?

6.请简述 LinkedHashMap 的工作原理和使用方式?

算法

1.什么是冒泡排序? 如何优化?

```
public class BubbleSort {
    public static void main(String[] args) {
        int [] arr = {6,2,4,7,5,1,9,8,3};
        BubbleSort(arr);
        System.out.println(Arrays.toString(arr));
    }
    public static void BubbleSort(int[] arr){
        int swap=0;
        for(int i = 0;i< arr.length-1;i++){
            for (int j = 0;j<arr.length-1-i;j++){
                if(arr[j]>arr[j+1]){
                   swap = arr[j];
                   arr[j]=arr[j+1];
                   arr[j+1]=swap;
                }
            }
        }
    }
}
```

2.请用 Java 实现一个简单的单链表?

```
public class Node {
    //当前结点的值
    public Integer value;
    //下一个结点
    public Node next;
    public Node(Integer value) {
        this.value = value;
    }
    public Node(Integer value, Node next) {
        this.value = value;
        this.next = next;
    }
    /* 省略getter和setter方法 */
}
```

3.如何反转一个单链表?

```
public class Solution {
```

```
    public ListNode ReverseList(ListNode head) {
//      如何调整链表指针，达到链表反转的目的。
        ListNode prev = null; // prev : 指向反转好节点的最后一个节点
        ListNode curr = head;    //指向反转链表的第一个节点
        while(curr != null){
            ListNode next = curr.next;
            curr.next = prev;

            prev = curr;
            curr = next;
        }
        return prev;
    }
}
```

4.谈谈你对时间复杂度和空间复杂度的理解?

5.谈一谈如何判断一个链表成环?

```
1.快慢指针的方法
两个指针一个fast一个slow 同时从头结点出发，先同时移动一次 然后fast再移动一次，两个指针相遇 即是链表成环
```

6.什么是红黑树? 为什么要用红黑树?

7.什么是快速排序? 如何优化?

```
public class QuickSort {

    public static void quickSort(int[] arr, int left, int right){
        if (left < right){
            // 把数组分块
            int pivot = partition(arr, left, right);
            System.out.println(Arrays.toString(arr));
            // 基准元素左边递归
            quickSort(arr, left, pivot-1);
            // 基准元素右边递归
            quickSort(arr, pivot+1, right);
        }
    }

    public static int partition(int[] arr,int left,int right){
        int pivot = arr[left];      //默认第一个元素是基准元素
        while (left<right){
            //右边往左移动 直到遇到小于基准元素的
            while(left < right && arr[right] >= pivot){
                right--;
            }
            arr[left] = arr[right]; //记录那个较小的值

            //左往右移动 直到遇到大于基准元素的
            while (left < right && arr[left] <= pivot){
                left++;
            }
            arr[right] = arr[left]; //记录那个较大的值
        }
        arr[left] = pivot;          //把基准元素放到当前指向的地方
        return left;                //返回基准元素的索引
    }
}

```

8.说说循环队列?

9.如何判断单链表交叉

计算机网络

1.请简述 Http 与 Https 的区别?

```
HTTP协议以明文方式发送内容，不提供任何方式的数据加密。HTTP协议不适合传输一些敏感信息，比如：信用卡号、密码等支付信息。https则是具有安全性的ssl加密传输协议。http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。并且https协议需要到ca申请证书。HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全。
```

2.说一说 https,udp,socket 区别?

```
https是在HTTP的应用层协议与TCP传输层协议之间套了一个安全套接层SSL/TLS
UDP与TCP也一样是一个传输层协议
Scoket是对TCP/IP协议的抽象，是操作系统对外开放的接口
```

3.请简述一次 http 网络请求的过程?

```
1、域名解析：使用DNS协议进行域名解析
2、建立连接：发起TCP三次握手
3、发起http请求：建立TCP连接成功后，浏览器发起http请求
4、响应http请求：服务端响应http请求，浏览器得到返回response
5、解析response：浏览器解析response，并请求其它的资源（如js、css等）
6、浏览器渲染展示页面：浏览器根据内核对页面进行渲染展示
7、断开连接：TCP四次挥手
```

4.谈一谈 TCP/IP 三次握手，四次挥手?

```
三次握手
用于保证可靠性和流量控制维护的某些状态信息，这些信息的组合，包括Socket、序列号和窗口大小称为连接。
三次握手才可以阻止重复历史连接的初始化（主要原因）
三次握手才可以同步双方的初始序列号
三次握手才可以避免资源浪费

四次挥手
第一次挥手客户端发起关闭连接的请求给服务端；
第二次挥手：服务端收到关闭请求的时候可能这个时候数据还没发送完，所以服务端会先回复一个确认报文，表示自己知道客户端想要关闭连接了，但是因为数据还没传输完，所以还需要等待；
第三次挥手：当数据传输完了，服务端会主动发送一个 FIN 报文，告诉客户端，表示数据已经发送完了，服务端这边准备关闭连接了。
第四次挥手：当客户端收到服务端的 FIN 报文过后，会回复一个 ACK 报文，告诉服务端自己知道了，再等待一会就关闭连接。
```

5.为什么说 Http 是可靠的数据传输协议?

6.TCP/IP协议分为哪几层？TCP 和 HTTP 分别属于哪一层?

```
tcp/ip协议包含应用层、传输层、网络层和数据链路层4层
TCP 在传输层
HTTP在应用层
IP再网络层
```

多线程

1.Java 中使用多线程的方式有哪些?

```
在java中实现多线程的两途径：继承Thread类，实现Runable接口（Callable）
```

2.说一下线程的几种状态?

3.如何实现多线程中的同步?

4.谈谈线程死锁，如何有效的避免线程死锁?

```
互斥条件：该资源任意一个时刻只由一个线程占用。
请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。
不剥夺条件:线程已获得的资源在未使用完之前不能被其他线程强行剥夺，只有自己使用完毕后才释放资源。
循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。

破坏互斥条件 ：这个条件我们没有办法破坏，因为我们用锁本来就是想让他们互斥的（临界资源需要互斥访问）。
破坏请求与保持条件 ：一次性申请所有的资源。
破坏不剥夺条件 ：占用部分资源的线程进一步申请其他资源时，如果申请不到，可以主动释放它占有的资源。
破坏循环等待条件 ：靠按序申请资源来预防。按某一顺序申请资源，释放资源则反序释放。破坏循环等待条件。
```

5.谈谈线程阻塞的原因?

6.请谈谈 Thread 中 run() 与 start() 的区别?

```
1) run()就和普通的成员方法一样，可以被重复调用。
2) start() 是开启一个新的线程，执行run方法，start方法不能重复调用。
```

7.synchronized和volatile关键字的区别?

8.如何保证线程安全?

9.谈谈ThreadLocal用法和原理?

10.Java 线程中notify 和 notifyAll有什么区别?

11.什么是线程池? 如何创建一个线程池?

```
线程池就是提前创建若干个线程,如果有任务需要处理,线程池里的线程就会处理任务,处理完之后线程并不会被销毁,而是等待下一个任务。
```

```java

public class ThreadPool1 {
    public static void main(String[] args) {
        //创建一个包含5个线程的线程池
        ExecutorService threadPool = Executors.newFixedThreadPool(5);
        //使用线程池执行任务
        for (int i = 0; i < 5; i++) {
            //可以用submit执行任务
            threadPool.submit(new Runnable() {
                @Override
                public void run() {
                    System.out.println("线程名: " + Thread.currentThread().getName());
                }
            });
        }
        for (int i = 0; i < 10; i++) {
            //也可以用execute执行任务
            threadPool.execute(new Runnable() {
                @Override
                public void run() {
                    System.out.println("线程名" + Thread.currentThread().getName());
                }
            });
        }
    }
}
```

12.谈一谈java线程常见的几种锁?

13.谈一谈线程sleep()和wait()的区别?

14.什么是悲观锁和乐观锁?

15.什么是BlockingQueue?请分析一下其内部原理并谈谈它的使用场景?

16.谈一谈java线程安全的集合有哪些?

17.Java中为什么会出现Atomic类?试分析它的原理和缺点?

18.说说ThreadLocal的使用场景?与Synchronized相比有什么特性?

设计模式

1.工作中常用的设计模式，一些源码中的设计模式

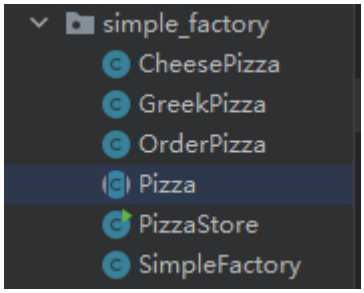
https://blog.csdn.net/qq_36386908/article/details/123416250

单例模式

```java
public class SingletonTestFinal {
    public static void main(String[] args) {
        Singleton instance1 = Singleton.INSTANCE;
        Singleton instance2 = Singleton.INSTANCE;
        System.out.println(instance1 == instance2);
    }
}

enum Singleton {
    INSTANCE;
}
```

工厂模式




```java
// 1.披萨抽象类
public abstract class Pizza {

    protected String name;

    public void setName(String name) {
        this.name = name;
    }

    public abstract void prepare();

    public void bake() {
        System.out.println(name + "baking;");
    }

    public void cut() {
        System.out.println(name + "cut;");
    }

    public void box() {
        System.out.println(name + "box;");
    }
}
// 2.希腊披萨
public class GreekPizza extends Pizza {
    @Override
    public void prepare() {
        System.out.println(name + "披萨准备原材料");
    }
}
// 3.奶酪披萨
public class CheesePizza extends Pizza {
    @Override
    public void prepare() {
        System.out.println(name + " 准备原材料");
    }
}
```

```java
//4.披萨工厂根据用户的而不同需求，生产不同类别的披萨
public class SimpleFactory {

    public Pizza createPizza(String pizzaType) {
        Pizza pizza = null;
        if("cheese".equals(pizzaType)) {
            pizza = new CheesePizza();
            pizza.setName("cheese");
        }else if("greek".equals(pizzaType)) {
            pizza = new GreekPizza();
            pizza.setName("greek");
        }
        return pizza;
    }
}
// 5.订购披萨
public class OrderPizza {

    private static Scanner scanner = new Scanner(System.in);
    private SimpleFactory simpleFactory;

    // 传入工厂
    public OrderPizza(SimpleFactory simpleFactory) {
        this.simpleFactory = simpleFactory;
    }

    public void getPizza() {
        while (true) {
            System.out.println("请输入披萨类型: ");
            // 从工厂获取披萨
            Pizza pizza = simpleFactory.createPizza(scanner.next());
            if (pizza != null) {
                pizza.prepare();
                pizza.bake();
                pizza.cut();
                pizza.box();
            } else {
                System.out.println("暂无该类型披萨！");
                break;
            }
        }
    }
}
// 6.购买披萨
public class PizzaStore {
    public static void main(String[] args) {
        // 创建披萨订购类，并传入披萨工厂
        OrderPizza orderPizza = new OrderPizza(new SimpleFactory());
        orderPizza.getPizza();
    }
}
```

3.具体给你一个设计模式让你说说你对他的了解，比如观察者，工厂。

以上这些东西主要考察你的代码设计能力。

简单工厂

> 优点

> 工厂类包含必要的逻辑判断，可以决定在什么时候创建哪一个产品的实例。客户端可以免除直接创建产品对象的职责，很方便的创建出相应的产品。工厂和产品的职责区分明确。
> 客户端无需知道所创建具体产品的类名，只需知道参数即可。
> 也可以引入配置文件，在不修改客户端代码的情况下更换和添加新的具体产品类
>
> 缺点
>
> 简单工厂模式的工厂类单一，负责所有产品的创建，职责过重，一旦异常，整个系统将受影响。且工厂类代码会非常臃肿，违背高聚合原则。
> 使用简单工厂模式会增加系统中类的个数（引入新的工厂类），增加系统的复杂度和理解难度
> 系统扩展困难，一旦增加新产品不得不修改工厂逻辑，在产品类型较多时，可能造成逻辑过于复杂
> 简单工厂模式使用了 static 工厂方法，造成工厂角色无法形成基于继承的等级结构

Android综合

1. 提一下RelativeLayout和LinearLayout的区别?

- **线性布局**- 垂直或水平排列元素。即在一行或一列中。
- **相对布局**- 相对于父元素或其他元素排列元素。

2. Android中Bitmap和Drawable有什么区别?

- **位图**是位图图像的表示（类似于 java.awt.Image）。
- **Drawable**是"可以绘制的东西"的抽象。它可以是位图（包装为 BitmapDrawable），但也可以是纯色、其他 Drawable 对象的集合或任意数量的其他结构。

3. Spannable 和 String 有什么区别?

Spannable允许将格式信息（如粗体、斜体、...）附加到字符的子序列（"跨度"，即名称）。只要您想表示"富文本"，就可以使用它。

4. 什么是Activity?

Activity提供了应用程序在其中绘制其 UI 的窗口。此窗口通常会填满屏幕，但可能会小于屏幕并浮动在其他窗口的顶部。通常，一个活动在应用程序中实现一个屏幕。例如，一个应用的 Activity 可能实现一个 Preferences 屏幕，而另一个 Activity 实现一个 Select Photo 屏幕。

5. 为什么建议只使用默认构造函数来创建 Fragment?

简而言之，Fragment 需要有一个无参数的构造函数供 Android 系统实例化它们。您的 Fragment 子类需要一个公共的空构造函数，因为这是框架调用的。

它用于设备必须恢复片段状态的情况。不会传递任何数据，会创建一个默认片段，然后恢复状态。由于系统无法知道您在构造函数或您的构造函数中传递了什么`newInstance`，因此将使用默认构造函数，并且在使用默认构造函数实际实例化片段后，应通过 onCreate 传递保存的包。

6. 如何在 Android 应用中持久化数据?

在 Android 应用程序中存储数据基本上有四种不同的方式：

1. Shared Preferences - 将原始数据保存在键值对中
2. 内部存储 - 您需要将数据存储到设备文件系统，但您不希望任何其他应用程序（甚至用户）读取此数据
3. 外部存储 - 您可能希望用户查看您的应用保存的文件和数据
4. SQLite 数据库

7.简述所有Android应用组件

应用程序组件是 Android 应用程序的基本构建块。每个组件都是一个入口点，系统或用户可以通过它进入您的应用程序。

有四种不同类型的应用程序组件：

- **活动**- 活动是与用户交互的入口点。它代表具有用户界面的单个屏幕。
- **服务**- 服务是一个通用入口点，用于出于各种原因让应用程序在后台运行。它是在后台运行以执行长时间运行的操作或为远程进程执行工作的组件。
- **广播接收器**- 广播接收器是一个组件，它使系统能够在常规用户流之外向应用程序传递事件，从而允许应用程序响应系统范围的广播公告。
- **内容提供者**- 内容提供者管理一组共享的应用程序数据，您可以将这些数据存储在文件系统、SQLite 数据库、Web 或您的应用程序可以访问的任何其他持久存储位置中。

8. 如何在Android应用程序的Activity之间传递数据?

问题

我有一个场景，通过登录页面登录后，每个活动都会有一个退出按钮。您能否指导我如何使会话 ID 可用于所有活动?

最简单的方法是将会话 ID 传递给您用于启动活动的**Intent中的注销活动**：

```
Intent intent = new Intent(getBaseContext(), SignoutActivity.class);
intent.putExtra("EXTRA_SESSION_ID", sessionId);
startActivity(intent);
```

访问下一个活动的意图：

```
String sessionId = getIntent().getStringExtra("EXTRA_SESSION_ID");
```

9. 什么是达尔维克?

Dalvik是一个即时 (JIT) 编译器。使用 JIT 一词，我们的意思是，每当您在移动设备上运行应用程序时，执行应用程序所需的那部分代码只会在那时编译，其余代码将在未来需要时。JIT 或 Just In Time 仅编译您的代码的一部分，并且它具有更小的内存占用，因此，它在您的设备上使用的物理空间非常少。

10. 解释活动生命周期

当用户浏览、离开和返回您的应用程序时，应用程序中的 Activity 实例会在其生命周期中通过不同的状态进行转换。

为了在活动生命周期的各个阶段之间导航转换，Activity 类提供了一组核心的六个回调：`onCreate()`、`onStart()`、`onResume()`、`onPause()`、`onStop()` 和 `onDestroy()`。当活动进入新状态时，系统会调用这些回调中的每一个。

11. 什么是 AsyncTask?

`AsyncTask` 是在 Android 中实现并行性的最简单方法之一，而无需处理像线程这样的更复杂的方法。尽管它提供了与 UI 线程的基本并行度，但它不应用于更长的操作（例如，不超过 2 秒）。

AsyncTask 有四种方法

- `onPreExecute()`
- `doInBackground()`
- `onProgressUpdate()`
- `onPostExecute()`

where `doInBackground()` 是最重要的，因为它是执行背景计算的地方。

12. 讲解Android中的构建过程

1. 第一步涉及使用 aapt（android 资产打包工具）工具编译资源文件夹 (/res)。这些被编译成一个名为 R.java 的类文件。这是一个只包含常量的类。
2. 第二步是将java源代码通过javac编译为.class文件，然后通过“dx”工具将class文件转换为Dalvik字节码，该工具包含在sdk“tools”中。输出是 classes.dex。
3. 最后一步涉及 android apkbuilder，它接受所有输入并构建 apk（android 打包密钥）文件。

13. 什么是ADB，它的用途是什么?

ADB是 Android Debug Bridge 的首字母缩写，它是 Android SDK（软件开发工具包）的一部分。它使用客户端-服务器-模型（即 adbd，ADB 守护程序，在设备上运行并且可以连接），并且在大多数情况下通过 USB 连接使用。也可以通过 WiFi（无线 adb）使用它。

您无需在 Android 设备上安装任何内容，因为 ADB 守护程序 (adbd) 已集成到 Android 操作系统中。它通常通过 PC 的命令行界面访问，其中安装了完整的 Android SDK（目前有几个 30 MB 的下载存档），或者为“非开发人员”提供了一个大规模精简的版本，有时称为“Mini ADB”或“ADB Essentials”（对于 Linux，这只是 adb 可执行文件；对于 Windows，它是 adb.exe 加上两个或三个 .dll 文件）。

14. onCreate() 和 onStart() 有什么区别?

- 该`onCreate()`方法在 Activity 生命周期中调用一次，无论是在应用程序启动时，还是在 Activity 被销毁然后重新创建时，例如在配置更改期间。
- 只要 Activity 对用户可见，就会调用该`onStart()`方法，通常是在`onCreate()`或之后`onRestart()`。

15. 什么情况下应该使用 RecyclerView 而不是 ListView?

RecyclerView是作为**ListView改进创建的，所以是的，您可以使用**ListView**控件创建附加列表，但使用**RecyclerView更容易，因为它：

- 在向上/向下滚动时重用单元格 - 这可以通过在 ListView 适配器中实现 View Holder 来实现，但它是可选的，而在 RecycleView 中它是编写适配器的默认方式。
- 将列表与其容器分离 - 因此您可以在运行时通过设置 LayoutManager 轻松地将列表项放入不同的容器（linearLayout、gridLayout）中。

总而言之，**RecyclerView**是一种更灵活的控件，用于处理“列表数据”，它遵循关注点委托的模式，只为自己留下一个任务——回收项目。