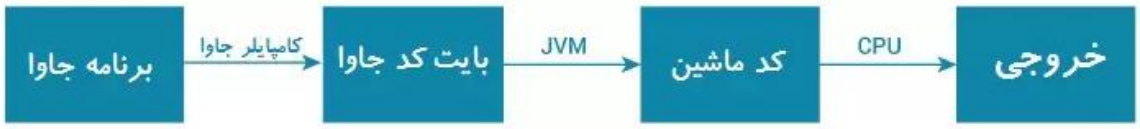
سوالات جاوا

1) تفاوت jvm , jdk, jre چیه ؟

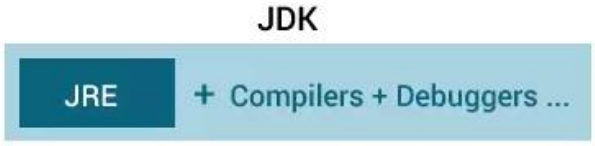
Jvm (ماشین مجازی جاوا – java virtual machine) یک ماشین انتزاعیه که سیستم ما میتونه به وسیله اون برنامه های جاوا رو اجرا کنه.

وقتی یه برنامه جاوا رو اجرا میکنید، کامپایلر جاوا ابتدا کدهای جاوا رو به بایت کد تبدیل میکنه بعدش jvm این بایت کد رو با کد ماشین تبدیل میکنه تا cpu بتونه دستورها رو اجرا بکنه

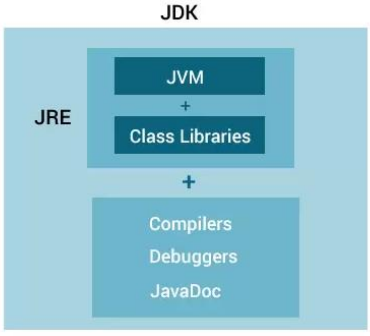


Jre (java runtime environment) یه پک نرم افزاریه که کتابخونه های کلاس جاوا رو به همراه jvm و بقیه مولفه ها برای اجرای برنامه های جاوا فراهم میکنه (اگر میخواید برنامه های جاوایی رو بدون develop کردن run کنید باید jre رو روی سیستم تون نصب کنید)

Jdk (java development kit – کیت توسعه ی جاوا) یه کیت توسعه نرم افزاری برا بسط و توسعه برنامه های جاویی هستش، وقتی شما jdk رو دانلود میکنید jre رو داخل خودش داره و نیازی به دانلود مجزا نیست . jdk علاوه بر jre یک سری ابزار توسعه مثل (کامپایلرها، java debug، java doc، ...) داخل خودش داره. (اگر میخواید برنامه های جاوایی رو develop کنید باید jdk رو توی سیستم تون نصب کنید)



رابطه بین jvm, jre, jdk رو توی عکس پایین نمایش دادیم:



2) چرا برای زبان جاوا میگن " یک بار بنویس همه جا اجرا کن" ؟
جاوا یه زبان مستقل از پلت فرمه ، یعنی وقتی شما کدی رو مینویسید، در اصل برای jvm اون کد رو مینویسید نه سیستم تون و از اونجایی که jvm بایت کد جاوا رو مستقل از سیستم عامل اجرا میکنه در نتیجه جاوا مستقل از پلت فرمه.

3) Jit compiler چیه؟
مفسر کدها رو به صورت خط به خط اجرا میکنه اما کامپایلر کل کدها رو به صورت یک جا کامپایل و اجرا میکنن یکی از بدی های مفسرها اینه که چون خط به خط کد ها رو بررسی و اجرا میکنن در بلاکه های لوپ معمولا این روند رو به طور مکرر انجام میدن ، jit compiler ها اومدن یه راهکاری ارائه کردن به این شکل که کدهای ما رو به اصطلاح monitor میکنن (یعنی تعداد اجرای هر کد و نحوه استفاده اون کد رو بررسی میکنن) اگه یک بخشی تعداد تکرارش بیشتر باشه jit به اون سگمن یک برچسب "گرم" میزنه و اون بخش رو کامپال میکنه و از اون به بعد کامپایل شده ی کد رو به مفسر برای اجرا تحویل میده، اگه بخشی بیش اندازه تکرار بشه برچسب "داغ" به اون میزنه که به جز کامپایل کردن سگمنت ، اون بخش رو از لحاظ کد نیز optimize(بهینه) نیز میکنه، حالا توی java وظیفه کامپایل کد توسط jvm انجام میشه jit در جاوا هنگام اجرا(run time) و بعد از هر بار اجرا کدها رو بررسی میکنه و در هر بار تکراتر کد اون رو بهینه میکنه تا در هنگام اجرا کدها بهینه تر عمل کنن

4) اگر به جای public static void بنویسم static public void چه اتفاقی میفته؟
برنامه به درستی کامپایل و اجرا می شود زیرا ترتیب مشخص کننده ها توی جاوا مهم نیست.

5) Default value در local variable ها چقدر است ؟
local variable ها default value ندارن و intilize نشدن، نه primitveها نه non primitive نه object reference ها ، باید حتما اونها رو با یک default value ای initialize کنیم ولی global value ها رو لازم نیست (چون از قبل به صورت رزور با یک default value ، initialize شده ن)

6) سطح دسترسی ها توی جاوا چیا هستن؟ public, protected, default, private
Public: تمامی کلاس ها و متدها و متغیرهایی که به صورت public تعریف میشن، توی هر کلاس و متدی قابل دسترسی هستش
Protected: توسط کلاس های همون package یا sub-class ش(بیرون یا داخل پکیج) یا همون کلاس قابل دسترسیه
Default: فقط درون پکیج قابل دسترسیه
Private: متدها و متغیرهایی که درون کلاس تعریف میشن فقط درون همون کلاس دسترسی دارن.

7) آیا میشه یهconstructor از جنس final داشت؟
نه

8) هدف از ایجاد static method ها و static variable ها چیه؟
متدها و متغیرهایی که static تعریف میشن بین تمامی اشیای کلاس اشتراک گذاری میشن، static ها بخشی از کلاس تلقی میشن نه object ها ، static variable ها توی ناحیه کلاس ذخیره میشن به خاطر همین برای دسترسی به این متغیرها نیازی به ایجاد object نداریم. بنابراین ، static ها زمانی استفاده میشن که باید متغیرها و متدهایی تعریف کنیم که برای همه ی اشیای کلاس مشترک باشن.
برای مثال ، کلاسی رو فرض کنید که مجموعه دانش آموزان یک مدرسه رو شبیه سازی میکنه، طبیعتا اسم مدرسه ویژگی مشترک همه ی دانش اموزان خواهد بود بنابرین باید اسم مدرسه رو به عنوان متغیر static تعریف کنیم.

9) چرا متد main استاتیک است؟
بخاطر اینکه برای صدا زدن متد استاتیک نیازی به ساختن شی وجود نداره ، اما اگر متد main غیر استاتیک بود اون وقت با هر بار اجرا کردن برنامه jvm باید ابتدا یک شی میساخت و بعد متد main رو صدا میزد که این باعث میشد محل حافظه زیادی الکی اشغال بشه.

10) Initial value یک object refrence که یه Inctance variable از اون تعریف میشه چقدره ؟
تمام object refrenceها مقدار اولیه شون Nullهست.
Instance variable : به variblae های سطح کلاس میگن(یعنی وقتی یه instance از یه کلاس میسازیم فقط به متغیرهای سطح کلاس دسترسی داریم و به متغیرهای درون متدها دسترسی نداریم به خاطر همین به متغیرهای سطح کلاس instance varible میگن)
اگه یه instance از یه کلاس بسازیم اون instance مقدارش null اما اگه اون instance رو intilize کنیم دیگه null نیست.
حالا این instance که Intilize کردیم اگه vriable ای داشته باشه اون هم null عه. مگر اینکه مقدار اولیه ای داده باشیم.

```
public class MiddleClass {
    3 usages
    Test test; // test از کلاس instance

    1 usage
    public void watch() {
        test = new Test(); // instance کردن از همون initilize
        System.out.println(test); // بود null خروجی مقدار نمیکردیم intilize
        System.out.println(test.one); //  instance variable is null
    }
```

11) میشه constructor رو به ارث برد؟
نه

12) میشه constructor رو final کرد؟
نه

13) Costructor رو میشه overload کرد؟
اره ، میشه constructor ها رو با تغییر تعداد ارگومان های ورودی overload کرد.

14) محدودیت های static methodها چیه؟
1- در static method ها نمیتونیم متغیرهای non-static (که در سطح کلاس تعریف میشن) رو call کنیم و تبعا وقتی نمیتونیم call کنیم پس تغییر و دستکاری اون غیر ممکنه.
2- از کلمات کلیدی this , super نمیتونیم توی static context ها استفاده کنیم چون non-static هستن.
3- متدهای non-static رو نمیشه توی متدهای static فراخونی کرد

15) آیا static methodها امکان override دارن؟
نه ، static method ها رو نمیشه override کرد

16) Static blockچیه؟
Static block ها برای intilize کردن متغیر های static استفاده میشه، stastic block ها قبل از main method اجرا میشه (در زمان classloading)

17) آیا امکان اجرای یک برنامه بدون متد main وجود داره؟
نه امکان پذیر نیست، قبل از jdk1.7 امکانش وجود داشت(با استفاده از static block ها) اما پس از اون این امکان برداشته شد.
18) اگر static رو از method main حذف کنیم چه اتفاقی میفته؟
برنامه کامپایل میشه اما در runtime خطای NoSuchMethodError میده

19) آیا static variable , static methodها رو میشه توی abstract class declare کرد؟
بله
20) کاربرد کلمه کلیدی this ؟
متدوال ترین کاربرد کلمه کلیدی this در جاوا برای فرق گذاشتن بین instance variable ها و local variable های هم نام.
21) میشه reference کلمه کلیدی this رو تغییر داد؟
نه چون کلمه this به object کلاسی که توی اون هستش اشاره میکنه (اشاره گر به کلاسی که در اون قرار داره) پس امکان تغییر reference وجود نداره
22) ایا از this میشه برای اشاره به staticها استفاده کرد؟
بله ، چون برای this فرقی نداره و در نهایت this یه reference variable برای اشاره به object کلاس فعلیه.
23) کدوم کلاس superclass ای برای همه کلاس هاست؟
Object class
24) inheritance چیه ؟
inheritance (وراثت یا ارث بری) مکانیزمی که بخاطر اون یک شی میتونه نمام property ها و عملکردهای یک شی دیگه توی یه کلاس دیگه رو بدست بیاره. از این قابلیت برای reusability و overriding استفاده میکنن، ایده پشت inheritance تو جاوا اینه که شما بتونی کلاسای جدیدی رو براساس کلاس های موجود بسازید. وقتی شما از یه کلاس ارث بری میکنید، میتونید از تمامی method ها و field های مربوط به parent کلاس تون استفاده کنید و علاوه بر اون میتونید method ها و field های جدیدی رو به کلاس فعلی تون اضافه کنید. پنج نوع راثت وجود داره:

1- single-level inheritance
2- multi-level inheritance
3- multiple inheritance
4- hierarchical inheritance
5- hybrid inheritance

جاوا multiple inheritance رو پشتیبانی نمیکنه.

25) چرا جاوا multiple inheritance رو پشتیبانی نمیکنه؟
برای کاهش پیچیدگی و ساده سازی این قابلیت پشتیبانی نمیشه، برای مثال سناریویی رو تصور کنید که در اون سه کلاس A,B,C وجود داره و کلاس C از کلاسهای A و B ارث بری میکنه؛ اگه جفت کلاس های A , B یه متد یکسان داشته باشند و کلاس C ارث بری کنه اون متد رو یه وضعیت گنگی توی call کردن متد از جفت کلاسای A,B اتفاق میفته ، که در این موقعیت برای multiple inheritance ارور کامپایل تایم میده.

26) aggregation چیست ؟
اگر یک کلاس، یک entity reference داشته باشد aggregation اتفاق افتاده که یک رابطه HAS-A است. برای مثال:

```
class Employee{
int id;
String name;
Address address;//Address is a class
}
```

کلاس Employee دارای فیلدهای متنوعی می باشد، یکی از فیلدهای آن Address می باشد که خود یک کلاس جداگانه است و دارای فیلدهای دیگری می باشد.

27) composition چیست؟

28) method overloading چیست ؟

method overloading یه تکنیک از polymorphism عه که به ما اجازه میده چندین متد مختلف با اسم های یکسان اما signature های مختلف داشته باشیم. دو روش برای method overloading وجود داره:

1- از طریق تغییر تعداد آرگومان های ورودی 2- از طریق تغییر نوع آرگومان های ورودی

method overloading باعث افزایش خوانایی و فهم بهتر و سریعتر برنامه میشه.

29) آیا overloading با تغییر return type متدها امکان پذیره؟ چرا؟

خیر ، برای جلوگیری از ایجاد ابهام در برنامه این کار امکان پذیر نیست و با انجام این حرکت ارور تایم دریافت میکنیم.

برای مثال :

```
class Adder{
static int add(int a,int b){return a+b;}
static double add(int a,int b){return a+b;}
}
class TestOverloading3{
public static void main(String[] args){
System.out.println(Adder.add(11,11));//ambiguity
}}
```

خروجی این برنامه به شکل زیر میباشد:

```
Compile Time Error: method add(int, int) is already defined in class Adder
```

30) آیا امکان overload متدهایی که static هستن وجود داره؟

امکان overload متدها فقط با اعمال کلمه کلیدی static به اونا امکان پذیر نیست (با فرض یکسان بودن نوع و تعداد آرگومان ها) اما اگه نوع یا تعداد آرگومان متدها متفاوت باشه این امکان وجود داره. برای مثال:

```
public class Animal
{
   void consume(int a)
   {
      System.out.println(a+" consumed!!");
   }
   static void consume(int a)
   {
      System.out.println("consumed static "+a);
   }
   public static void main (String args[])
   {
      Animal a = new Animal();
      a.consume(10);
      Animal.consume(20);
   }
}
```

در این مثال برنامه خطا میده اما اگه یکی از متدها نوع یا تعداد آرگومان متفاوت باشه، برنامه بدون مشکل اجرا میشه.

31) ایا امکان overload متد main وجود داره؟
بله میشه به هر تعدادی که میخواهیم متد main رو overload کنیم.
32) overriding چیست؟
وقتی که کلاس فرزند متد کلاس پدر رو توی بدنه خودش پیاده سازی کنه و دستورات درون بدنه اون رو تغییر بده ، به این عمل Override کردن متد کلاس پدر در کلاس فرزند میگیم. این تغییرات تنها شامل دستورات درون بدنه متد میشه و حق تغییر نوع پارامترها و نام متد رو نداریم.
33) آیا static method رو میشه override کرد؟
نه نمیشه ، چون static بخشی از class هستش نه object .
34) آیا میشه متد overload شده رو override کرد؟
بله
35) آیا private method ها رو میشه override کرد؟
نه ، چون متد های private محدود به کلاسی که توی اون قرار دارن هستن و اجازه دسترسی بیرون کلاس مربوطه امکان پذیر نیستش.
36) آیا سطح دسترسی متد override شده رو میشه توی subclass تغییر داد؟
بله میشه ، اما نمیشه سطح دسترسی رو کمتر کرد و با توجه به این نکته به سه شکل میشه دسترسی رو تغییر داد:
private → public, protected, default
protected → public, default
default → public
37) Can we modify the throws clause of the superclass method while overriding it in the subclass?
38) final variable چیست ؟
توی جاوا final variable برای محدود کردن تغییر variable استفاده میشه یعنی اگه ما یه final variable رو initialize کنیم دیگه value اون رو جای دیگه ای از برنامه نمیتونیم تغییر بدیم.
39) final method چیست؟
اگه یه متدی به صورت final تعریف بشه دیگه نمیشه اون متد رو override کرد.
40) final classچیست؟
اگه شما یک کلاس رو final کنید دیگه نمیتونید از اون ارث بری کنید.
41) آیا متد final رو میشه ارث بری کرد ؟
بله ، اما نمیتونیم override کنیم.
42) ایا امکان تعریف متغیر final بدون مقداردهی امکان پذیر است؟
بله ، تنها در یه صورت میشه متغیر final رو مقداردهی یا به عبارت دیگه initialize نکرد که اون متغیر رو در constructor مقداردهی کنیم.
43) با توجه به سوال قبل اگه متغیر static هم باشه چی؟
اگه متغیر به جز final ، static هم باشه اون وقت امکان مقداردهی در constructor وجود نداره و باید برای Initilize کردن یه static block تعریف کنیم.
44) آیا main method رو میشه final کرد؟
بله (public static final void main(String[] args))
45) آیا میشه یه constructor رو final کرد؟
خیر نمیشه.
46) آیا میشه یه interfaceرو finalکرد ؟
خیر نمیشه. یه Interfaceبرای اینکه تعریف بشه باید توسط یه کلاس implement بشه پس هیچ معنایی نداره که بخواد final باشه.

47) runtime polymorphism چیست ؟

فرایندی که متدهای override شده توی runtime بررسی میشن نه compile time ، توی این فرایند متد override شده توسط refrence varible ای که از جنس superclass هست call میشه . برای درک بیشتر مثال زیر رو ببینی:

```
1. class Bike{
2.   void run(){System.out.println("running");}
3. }
4. class Splendor extends Bike{
5.   void run(){System.out.println("running safely with 60km");}
6.   public static void main(String args[]){
7.     Bike b = new Splendor();//upcasting
8.     b.run();  //output → running safely with 60km
9.   } }
```

توی این مثال variable b از جنس سوپر کلاس (Bike) هست اما از جنس Splendor ، new شده (مفهوم upcasting) و متد override شده مقدار درون کلاس Splendore رو نمایش میده که این در زمان اجرا run time بررسی میشه .

48) در زمان runtime polymorphism ، varibleهای subclass صدا زده میشه یا superclass ؟

superclass ، وقتی از upcasting استفاده میکنیم چون جنس object از جنس superclass هست پس با new کردن از جنس subclass ، object به متد های override شده از superclass و متغیرهای مربوط به superclass دسترسی داره . مثال :

```
1. class Bike{
2.  int speedlimit=90;
3. }
4. class Honda3 extends Bike{
5.  int speedlimit=150;
6.  public static void main(String args[]){
7.   Bike obj=new Honda3();
8.   System.out.println(obj.speedlimit);// output → 90
9.    }
```

49) عملگر instanceOf در java چه کاربردی دارد؟

برای مقایسه نوع object از این عملگر استفاده میشه که مقدار true یا false برمیگردونه. مثال :

```
1. class Simple1{
2.  public static void main(String args[]){
3.  Simple1 s=new Simple1();
4.  System.out.println(s instanceof Simple1);//true
```

50) تعریف abstraction ؟

مراجعه به pdf جاوا پرو

51) یه متد رو میشه هم abstract و هم final تعریف کرد؟

نه ، چون وقتی یه متد رو abstract میکنیم ، میخاویم اون رو توی subclass ، override کنیم در صورتیکه وقتی یه متد final باشه نمیشه اون رو override کرد.

52) ایا از abstract میشه object ساخت؟

خیر

53) interface چیست ؟

interface مانند یک کلاس بنظر می رسه اما یک کلاس نیست!!! یک interface مثل کلاس می تونه متغیر و متد داشته باشد با این تفاوت که متدهای درون یک اینترفیس مثل متدهای کلاس abstract بدنه ندارند.یعنی متدهای یک interface دارای نوع ، نام ، پارامتر هستند اما بدنه ندارند .در interface متغیرها می توانند public ، static، final، defaultاعلام و تعریف شوند .تمامی متدهای درون Interface بدون بدنه هستند.به عبارتی می توان گفت که اینترفیس ها مجموعه ای از متدهای abstract هستند

54) آیا یه متد static در یه کلاس interface میشه تعریف کرد؟

اگر متدی که مینویسیم بدون بدنه باشه امکان static کردن وجود نداره اما اگه متد دارای بدنه باشه امکان static تعریف کردن متد وجود داره.

55) آیا اماکن final کردن interfaceوجود داره؟

نه چون وقتی یه کلاس رو interface تعریف میکنیم قراره توسط کلاس های دیگه implement بشن و اگه کلاس متدهاش final باشه امکان implement وجود نداره.

56) امکان تعریف متغیرهای private و protected و default در interface وجود دارد؟

خیر، فقط public

57) چطور میشه یه کلاس read-only یا write-only تعریف کرد؟

تنها در صورتی امکان پذیر است که تمام تغیرها به صورت private تعریف بشن و برای ایجاد کلاس read-only برای تمام متغیرها getter تعریف بشه و برای حالت write-only ، setter .

58) چند مدل exception توی جاوا داریم؟

به طور کلی دو مورد : checked , unchecked و error جزو unchecked محسوب میشه اما در ساختار oracle سه مورد : checked, unchecked, error

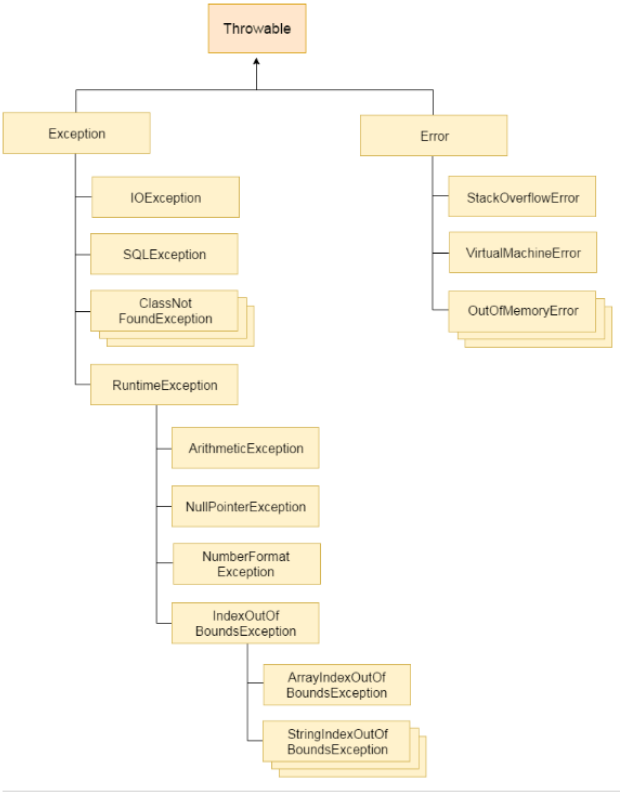
checked : به اررورهای compile-time میگن مثل : IOException , SQLException

unchecked : به اراوهای run-time میگن مثل : ArithmaticException, NullPointerException, ArrayIndexOutOfBoundsException

error : خطاهایی که باعث خروج از برنامه میشن چون قابل بازیابی نیستن. مثل : OutOfMemoryError, AssertionError

ساختار سلسله مراتبی exception ها :

نکته : کلاس throwable به عنوان کلاس root برای exception ها محسوب میشه .



59) تفاوت throw با throws چیه ؟

Throws برای کنترل و مدیریت استثنا استفاده میشه و Throw برای تولید استثنا استفاده می شود.

60) وضعیت قرارگیری try catch finally ؟

یک بلوک try میتواند چندین بلوک catch داشته باشد ولی فقط یک بلوک finally دارد

بلوک finally بعد از بلوک catch میاد

یک بلوک catch بدون دستور try نمی تواند وجود داشته باشد

بلوک try می تواند بدون catch و تنها با بلوک finally پیاده سازی شود

بلوک try نمی تواند بدون بلوک catch یا finally پیاده سازی شود

هیچ کدی نمی تواند در بین بلوک هایtry ، catchو finally قرار بگیره

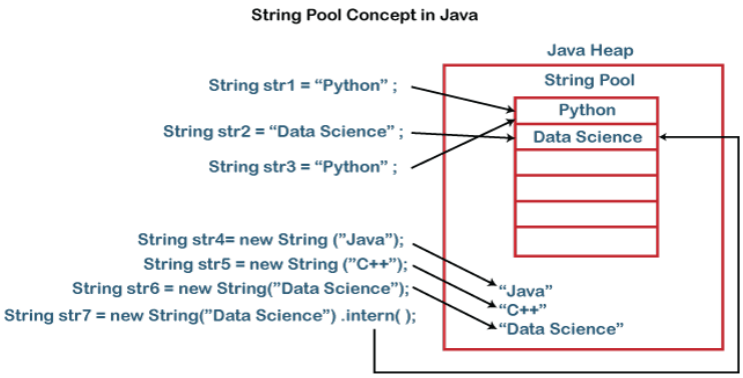
از نسخه JDK 7 به بعد شما می توانید چندین کالس استثنا را در یک بلوک catch تعریف کنید.با این کار دیگر شما نیاز به تعریف چندین بلوک catch نخواهید داشت مثال :

```
try {
      System.out.println("Access element zero Array a:" + a[0]);
      System.out.println("Access element one Array a/0:" + a[1] / 0);
      System.out.println("Access element two Array a:" + a[2]);

} catch (ArrayIndexOutOfBoundsException | ArithmeticException e) {
      System.err.println(e.getMessage());
      System.err.println(e + "\n");
}
```

61) string pool چیه ؟ string pool چیزی نیست جز یک فضای ذخیره سازی در heap memory. درست مثل تخصیص حافظه به object. به طور پیش فرض خالیه. هر زمان که رشته ای ایجاد می کنیم، object مورد نظر مقداری فضا در heap memory اشغال می کند. ایجاد تعدادی رشته ممکن است هزینه بر باشد و ممکن است عملکرد را هم کاهش دهد.

درنتیجه JVM مراحلی را در طول مقداردهی اولیه رشته انجام می دهد که باعث افزایش کارایی و کاهش بار حافظه می شود. برای کاهش تعداد اشیاء رشته ایجاد شده در JVM، کلاس String مجموعه ای از رشته ها را نگه می دارد. هنگامی که ما یک رشته ایجاد می کنیم، JVM ابتدا آن را در String Pool بررسی می کند. اگر از قبل در String Pool وجود داشته باشد، تنها نمونه یا همان reference را برمی گرداند در غیر این صورت، یک شی رشته جدید درString Pool ایجاد میشود. مثال :

```
String str1 = "Python"
String str2 = "Data Science"
String str3 = "Python"

s1==s3 //true
s2==s3 //false
```



ابتدا، ما یک رشته Python ایجاد کردیم و در String Pool جای گرفت. پس از آن، رشته Data Science ایجاد می شود، همچنین در String Pool قرار می گیرد. در نهایت، دوباره رشته Python را ایجاد کردیم. اما در این زمان، JVM رشته را بررسی می‌کند و متوجه می‌شود که رشته Python از قبل وجود دارد. به جای ساختن یک نمونه جدید در String Pool مرجع (reference) آن، یعنی str1 را برمی گرداند. و حالا یک نمونه مثال دیگر با کلید واژه new :

```
String str1 = new String ("Java");
String str2 = new String ("C++");
String str3 = new String ("Data Science");
```

رشته های Java, C++, Data Science ایجاد شدند. رشته های Java, C++ جدید هستند. اما Data Science در String Pool وجود دارد. همانطور که در عکس بالا دیدید تمام رشته های ایجاد شده با کلید واژه new در مموری هیپ جای می گیرند، نه در String Pool.

به مثال زیر توجه کنید :

```
public static void main(String[] args){
String s1 = "Java"
String s2 = "Java"
String s3 = new String("Java");
String s4 = new String("Java").intern();
System.out.println((s1 == s2)); // true
System.out.println((s1 == s3)); // false
System.out.println((s1 == s4)); // true
System.out.println((s2 == s4));//true
System.out.println((s3 == s4)); //false
System.out.println((s2 == s3)); //false
}
```

در مثال بالا میبینیم هر زمان که رشته ای را با کلید واژه new ایجاد کردیم، یک شی جدیدی در مموری هیپ ایجاد شده است. ما می‌توانیم با استفاده از متد intern کلاس String این ویژگی را متوقف کنیم.

نکته: برای هر رشته str1 و str2 عبارت str1.intern() == str2.intern تنها در صورتی true برمیگرداند که str1.equals(str2)، مقدار true برگرداند.

62) تفاوت string با string buffer و string builder ؟

63) nested class چیه ؟

اگه یه کلاس یا اینترفیس درون کلاس دیگه ای تعریف بشه به اون nested class میگن که شامل دو مدله : اول static nested class و دوم non-static nested class که به این مورد Inner class هم میگن

64) انواع inner class ها ؟

member inner class : کلاسی که دورن کلاس دیگر و بیرون از متدهای کلاس بیرونی تعریف میشه. مثال :

```
1.  class TestMemberOuter1{
2.   private int data=30;
3.   class Inner{
4.    void msg(){System.out.println("data is "+data);}
5.   }
6.   public static void main(String args[]){
7.    TestMemberOuter1 obj=new TestMemberOuter1();
8.    TestMemberOuter1.Inner in=obj.new Inner();
9.    in.msg();
10.  }
11. }
```

anonymous inner class : کلاسی که برای implement یه interface یا extend کردن از یه کلاس تعریف میشه. مثال:

```
1.  abstract class Person{
2.    abstract void eat();
3.  }
4.  class TestAnonymousInner{
5.   public static void main(String args[]){
6.    Person p=new Person(){
7.    void eat(){System.out.println("nice fruits");}
8.    };
9.    p.eat();
10.  }
11. }
```

local inner class : کلاسی که داخل متد تعریف میشه. مثال :

```
1.  public class localInner1{
2.   private int data=30;//instance variable
3.   void display(){
4.    class Local{
5.     void msg(){System.out.println(data);}
6.    }
7.    Local l=new Local();
8.    l.msg();
9.   }
10.  public static void main(String args[]){
```

```
11.     localInner1 obj=new localInner1();
12.     obj.display();
13.   }
14.  }
```

65) به local variable ها داخل local inner class میشه دسترسی داشت ؟

بله ، امکان دسترسی به local variable ها وجود داره اما امکان تغییرشون وجود نداره.

66) سوال بالا برای instance variable ها به چه شکلی هست؟

امکان دسترسی و تغییرشون موجود داره.

```java
public class A {

    int testA = 10;

    void testmethod() {
        int value = 50;
        class local {
            void message() {
                testA = 20; // دارای دسترسی و قابل تغییر
                value = 40; // دارای دسترسی ولی غیر قابل تغییر
                System.out.println(value + testA);
            }
        }
    }


}
```

67) interface میتونه یه class داخل خودش داشته باشه؟

بله به شرطی که کلاس از جنس static باشه.

68)