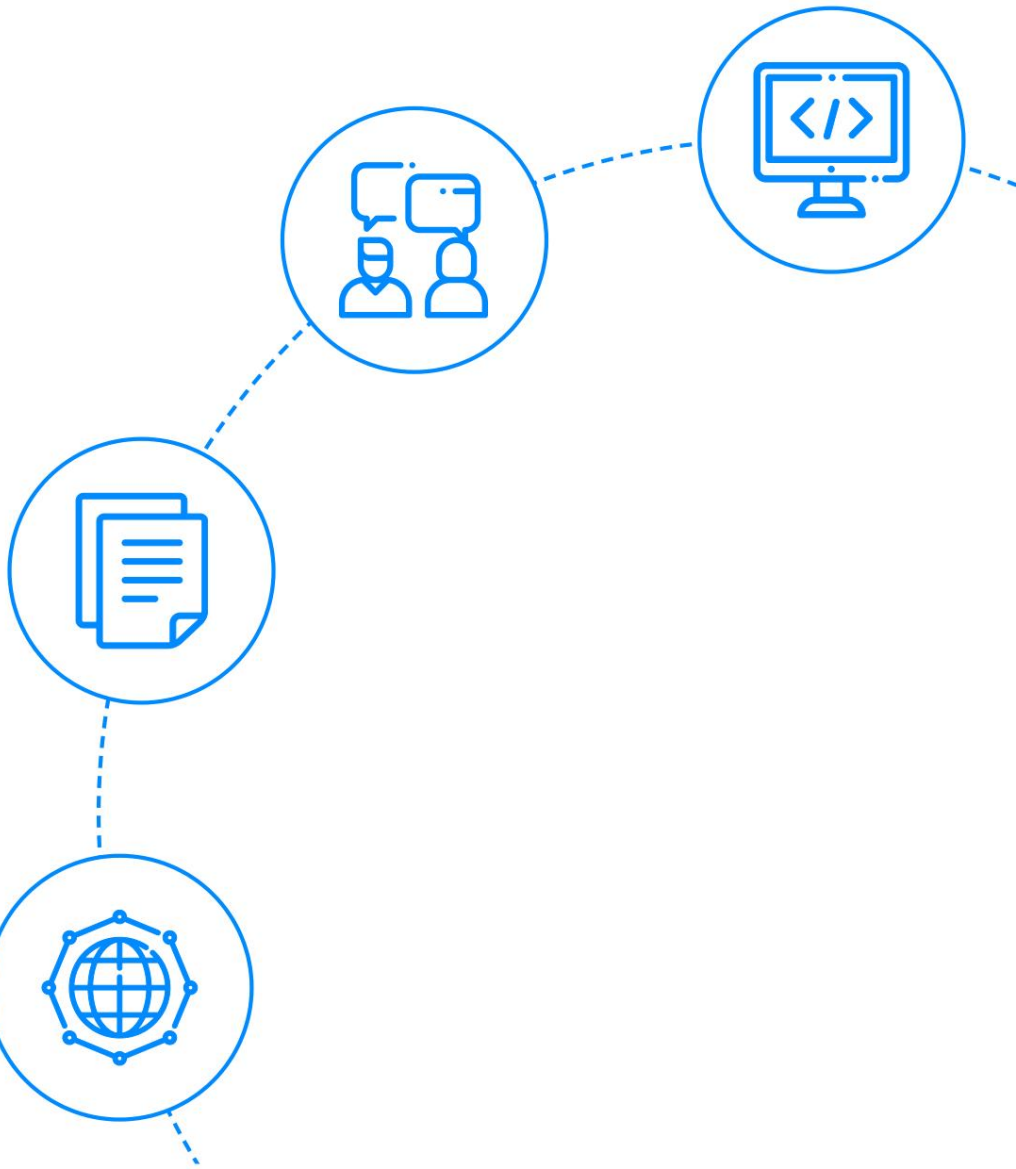
InterviewBit

जावा 8 साक्षात्कार प्रश्न



का लाइव संस्करण देखने के लिए
पेज, यहां क्लिक करें।

अंतर्वस्तु

जावा 8 फ्रेशर्स के लिए साक्षात्कार प्रश्न

अनुभवी के लिए जावा 8 साक्षात्कार प्रश्न

जावा 8 साक्षात्कार प्रश्न

अनुभव

(..... जारी)

19. वैकल्पिक वर्ग का उपयोग करने के क्या लाभ हैं?
20. जावा 8 स्ट्रीम क्या हैं?
21. धारा के मुख्य घटक क्या हैं?
22. डेटा ऑब्जेक्ट के स्रोत क्या हैं जो एक स्ट्रीम संसाधित कर सकता है?
23. इंटरमीडिएट और टर्मिनल ऑपरेशन क्या हैं?
24. सबसे अधिक उपयोग किए जाने वाले इंटरमीडिएट ऑपरेशन क्या हैं?
25. स्टेटफुल इंटरमीडिएट ऑपरेशन क्या है? स्टेटफुल के कुछ उदाहरण दें
    मध्यवर्ती संचालन।
26. टर्मिनल संचालन का सबसे सामान्य प्रकार क्या है?
27. फाइंडफर्स्ट () और फाइंडएनी () में क्या अंतर है?
28. संग्रह, स्ट्रीम से किस प्रकार भिन्न हैं?
29. जावा 8 में नई तिथि और समय एपीआई की विशेषता क्या है?
30. नए डेटा और टाइम एपीआई के लिए महत्वपूर्ण पैकेज क्या हैं?
31. उदाहरण के साथ समझाएं, लोकलडेट, लोकलटाइम और लोकलडेटटाइम एपीआई।
32. जावा में नैशॉर्न को परिभाषित करें 8
33. जावा 8 में जेजेएस का क्या उपयोग है?

Let's get Started

Java, originally evolved from the Oak language, was born in early 1996 with its major version as Java 1 or JDK 1.0. Java was initially designed and developed by Sir James Gosling at Sun Microsystems. Java 8 or JDK 8.0 is one of the major releases of the Java programming language in 2014. It is also known by the codename Spider. Java is an open-source project and is currently managed by Oracle Corporation.

This article would walk you through the Java 8 interview questions for freshers and experienced, scope, and opportunities.

Java 8 Interview Questions for Freshers

1. Describe the newly added features in Java 8?

Here are the newly added features of Java 8:

फ़ीचर का नाम	विवरण
लैम्ब्डा अभिव्यक्ति	एक फ़ंक्शन जिसे साझा किया जा सकता है या किसी ऑब्जेक्ट के रूप में संदर्भित किया जा सकता है।
कार्यात्मक इंटरफेस	एकल सार विधि इंटरफ़ेस।
तरीका संदर्भ	एक विधि को लागू करने के लिए एक पैरामीटर के रूप में फ़ंक्शन का उपयोग करता है।
डिफ़ॉल्ट विधि	यह 'इंटरफ़ेस विकास' सुविधाओं को सक्षम करने वाले इंटरफेस के भीतर विधियों का कार्यान्वयन प्रदान करता है।
स्ट्रीम एपीआई	सार परत जो डेटा की पाइपलाइन प्रसंस्करण प्रदान करती है।
दिनांक समय एपीआई	नए बेहतर जोडा-टाइम ने एपीआई को पिछले संस्करणों में कमियों को दूर करने के लिए प्रेरित किया
वैकल्पिक	रैपर वर्ग शून्य मानों की जांच करता है और मूल्य के आधार पर आगे की प्रक्रिया में मदद करता है।
नैशोर्न, जावास्क्रिप्ट यन्त्र	जावास्क्रिप्ट इंजन का एक सुधारित संस्करण जो राइनो को बदलने के लिए जावा में जावास्क्रिप्ट निष्पादन को सक्षम बनाता है।

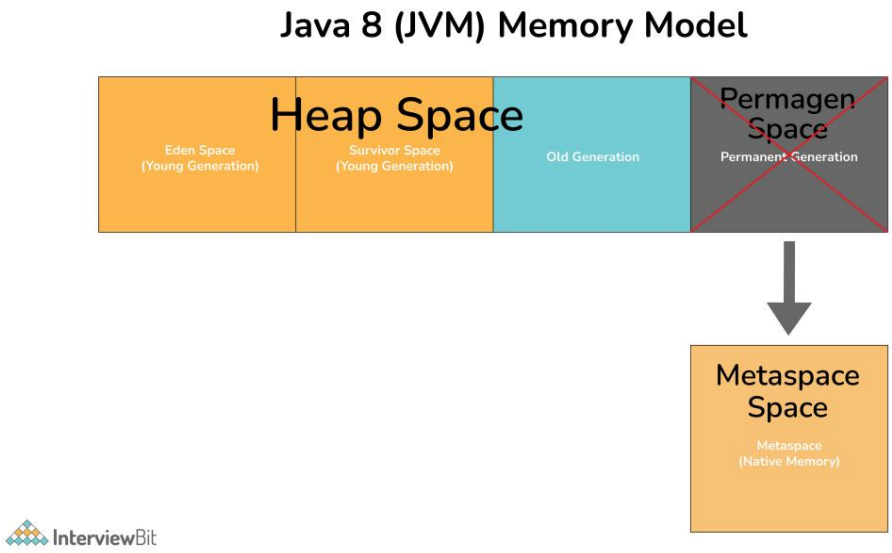
2. जावा 8 किस प्रोग्रामिंग प्रतिमान में आता है?

- वस्तु-उन्मुख प्रोग्रामिंग भाषा।
- कार्यात्मक प्रोग्रामिंग भाषा।
- प्रक्रियात्मक प्रोग्रामिंग भाषा।
- तर्क प्रोग्रामिंग भाषा

3. जावा 8 के महत्वपूर्ण लाभ क्या हैं?

- कॉम्पैक्ट, पठनीय और पुन: प्रयोज्य कोड।
- कम बॉयलरप्लेट कोड।
- समानांतर संचालन और निष्पादन।
- ऑपरेटिंग सिस्टम में पोर्ट किया जा सकता है।
- उच्च स्थिरता।
- स्थिर वातावरण।
- पर्याप्त समर्थन

4. मेटास्पेस क्या है? यह पर्मजेन से किस प्रकार भिन्न है?



जेवीएम

PremGen: कक्षाओं की मेटाडेटा जानकारी जावा 8 से पहले PremGen (स्थायी पीढ़ी) मेमोरी प्रकार में संग्रहीत की गई थी। PremGen आकार में निश्चित है और गतिशील रूप से आकार नहीं बदला जा सकता है। यह एक सन्निहित जावा हीप मेमोरी थी।

MetaSpace: Java 8 stores the MetaData of classes in native memory called 'MetaSpace'. It is not a contiguous Heap Memory and hence can be grown dynamically which helps to overcome the size constraints. This improves the garbage collection, auto-tuning, and de-allocation of metadata.

5. What are functional or SAM interfaces?

Functional Interfaces are an interface with only one abstract method. Due to which it is also known as the Single Abstract Method (SAM) interface. It is known as a functional interface because it wraps a function as an interface or in other words a function is represented by a single abstract method of the interface.

Functional interfaces can have any number of default, static, and overridden methods. For declaring Functional Interfaces @FunctionalInterface annotation is optional to use. If this annotation is used for interfaces with more than one abstract method, it will generate a compiler error.

```
@FunctionalInterface // एनोटेशन वैकल्पिक सार्वजनिक इंटरफ़ेस है फू () {

// डिफ़ॉल्ट विधि - वैकल्पिक 0 या अधिक सार्वजनिक डिफ़ॉल्ट स्ट्रिंग हैलोवर्ल्ड ()
{ वापसी "हैलो वर्ल्ड" हो सकती है; }


// स्टेटिक विधि - वैकल्पिक 0 या अधिक सार्वजनिक स्थैतिक स्ट्रिंग कस्टममैसेज (स्ट्रिंग
संदेश) { वापसी संदेश हो सकता है ; }


// एकल सार विधि सार्वजनिक शून्य बार (); }



सार्वजनिक वर्ग FooImplementation फू लागू करता है { // डिफ़ॉल्ट विधि - ओवरराइड
करने के लिए वैकल्पिक @ ओवरराइड सार्वजनिक डिफ़ॉल्ट स्ट्रिंग हैलोवर्ल्ड () { वापसी "हैलो
जावा 8"; }



// विधि ओवरराइड @ ओवरराइड
सार्वजनिक शून्य बार ()
{System.out.println ("हैलो
        वर्ल्ड");

}}

सार्वजनिक स्थैतिक शून्य मुख्य (स्ट्रिंग [] तर्क) {

फूइम्प्लीमेंटेशन फाई = नया फूइम्प्लीमेंटेशन (); System.out.println (fi.HelloWorld
()); System.out.println (fi.CustomMessage ("हाय")); फाई.बार (); }
```

6. क्या एक कार्यात्मक इंटरफ़ेस दूसरे इंटरफ़ेस का विस्तार/विरासत कर सकता है?

एक कार्यात्मक इंटरफ़ेस अमूर्त विधियों के साथ दूसरे इंटरफ़ेस का विस्तार नहीं कर सकता क्योंकि यह प्रति कार्यात्मक इंटरफ़ेस में एक सार विधि के नियम को शून्य कर देगा। जैसे:

```
इंटरफ़ेस अभिभावक { सार्वजनिक
int parentMethod (); }

@FunctionalInterface // यह फंक्शनलइंटरफेस इंटरफ़ेस नहीं हो सकता है चाइल्ड पैरेंट को बढ़ाता है { public int childMethod ();


// यह पेरेंट इंटरफेस की अमूर्त विधि का भी विस्तार करेगा
// इसलिए इसमें एक से अधिक अमूर्त विधियाँ होंगी
// और एक कंपाइलर त्रुटि देगा }
```

यह अन्य इंटरफेस का विस्तार कर सकता है जिसमें कोई सार विधि नहीं है और केवल डिफ़ॉल्ट, स्थिर है, एक और वर्ग ओवरराइड है, और सामान्य विधियां हैं। उदाहरण के लिए:

```
इंटरफ़ेस पैरेंट { सार्वजनिक शून्य
parentMethod () { System.out.println ("हैलो"); } }


@FunctionalInterface इंटरफ़ेस चाइल्ड
माता-पिता का विस्तार करता है { public int childMethod (); }
```

7. डिफ़ॉल्ट विधि क्या है, और इसकी आवश्यकता क्यों है?

इंटरफ़ेस में एक विधि जिसमें पूर्वनिर्धारित निकाय होता है उसे डिफ़ॉल्ट विधि के रूप में जाना जाता है। यह डिफ़ॉल्ट कीवर्ड का उपयोग करता है। जावा 8 में डिफ़ॉल्ट विधियों को पेश किया गया था 'यदि जेडीके किसी इंटरफेस को संशोधित करता है तो पिछड़ा संगतता'। यदि इंटरफ़ेस में एक नई अमूर्त विधि जोड़ी जाती है, तो इंटरफ़ेस को लागू करने वाले सभी वर्ग टूट जाएंगे और नई विधि को लागू करना होगा। डिफ़ॉल्ट विधियों के साथ, इंटरफ़ेस लागू करने वाली कक्षाओं पर कोई प्रभाव नहीं पड़ेगा। कार्यान्वयन में आवश्यक होने पर डिफ़ॉल्ट विधियों को ओवरराइड किया जा सकता है। इसके अलावा, यह सिंक्रनाइज़ या अंतिम के रूप में योग्य नहीं है।

```
@FunctionalInterface // एनोटेशन वैकल्पिक सार्वजनिक इंटरफ़ेस है फू () {

// डिफ़ॉल्ट विधि - वैकल्पिक 0 या अधिक सार्वजनिक डिफ़ॉल्ट स्ट्रिंग हैलोवर्ल्ड ()
{ वापसी "हैलो वर्ल्ड" हो सकती है; }


// एकल सार विधि सार्वजनिक शून्य बार (); }
```

8. इंटरफेस में स्थिर तरीके क्या हैं?

स्थैतिक विधियाँ, जिनमें विधि कार्यान्वयन शामिल है, इंटरफ़ेस के स्वामित्व में है और इंटरफ़ेस के नाम का उपयोग करके लागू की जाती है, यह उपयोगिता विधियों को परिभाषित करने के लिए उपयुक्त है और इसे ओवरराइड नहीं किया जा सकता है।

9. कुछ मानक जावा पूर्व-परिभाषित कार्यात्मक इंटरफेस क्या हैं?

पिछले जावा संस्करणों के कुछ प्रसिद्ध पूर्व-परिभाषित कार्यात्मक इंटरफेस रननेबल, कॉल करने योग्य, तुलनित्र और तुलनात्मक हैं। जबकि जावा 8 प्रदायक, उपभोक्ता, विधेय, आदि जैसे कार्यात्मक इंटरफेस पेश करता है। कृपया अन्य पूर्वनिर्धारित कार्यात्मक इंटरफेस और जावा 8 में पेश किए गए इसके विवरण के लिए java.util.function दस्तावेज़ देखें।

रननेबल: किसी अन्य थ्रेड पर किसी वर्ग के उदाहरणों को निष्पादित करने के लिए उपयोग करें जिसमें कोई तर्क नहीं है और कोई वापसी मूल्य नहीं है।

कॉल करने योग्य: किसी अन्य थ्रेड पर किसी वर्ग के उदाहरणों को बिना किसी तर्क के निष्पादित करने के लिए उपयोग करें और यह या तो एक मान देता है या अपवाद फेंकता है।

तुलनित्र: उपयोगकर्ता द्वारा परिभाषित क्रम में विभिन्न वस्तुओं को क्रमबद्ध करने के लिए उपयोग करें

तुलनीय: वस्तुओं को प्राकृतिक क्रम में क्रमबद्ध करने के लिए उपयोग करें

10. पूर्व-परिभाषित फ़ंक्शन इंटरफेस की विभिन्न श्रेणियां क्या हैं?

फ़ंक्शन: तर्कों को वापसी योग्य मान में बदलने के लिए।

विधेय: एक परीक्षण करने के लिए और एक बूलियन मान वापस करने के लिए।

उपभोक्ता: तर्क स्वीकार करें लेकिन कोई मान वापस न करें।

प्रदायक: किसी भी तर्क को स्वीकार न करें लेकिन एक मूल्य लौटाएं।

ऑपरेटर: एक कमी प्रकार का ऑपरेशन करें जो समान इनपुट प्रकारों को स्वीकार करता है।

11. जावा में लैम्ब्डा एक्सप्रेशन क्या है और लैम्ब्डा एक्सप्रेशन एक कार्यात्मक इंटरफ़ेस से कैसे संबंधित है?

लैम्ब्डा एक्सप्रेशन बिना नाम के एक प्रकार का फंक्शन है। इसके परिणाम और पैरामीटर हो सकते हैं या नहीं भी हो सकते हैं। इसे एक अनाम फ़ंक्शन के रूप में जाना जाता है क्योंकि इसमें अपने आप में प्रकार की जानकारी नहीं होती है। इसे ऑन-डिमांड निष्पादित किया जाता है। यह संग्रह से डेटा को पुनरावृत्त करने, फ़िल्टर करने और निकालने में फायदेमंद है।

चूंकि लैम्ब्डा अभिव्यक्ति अज्ञात कार्यों के समान हैं, इसलिए उन्हें केवल कार्यात्मक इंटरफ़ेस की एकल सार विधि पर लागू किया जा सकता है। यह कार्यात्मक इंटरफ़ेस की अमूर्त विधि के हस्ताक्षर से रिटर्न प्रकार, प्रकार और कई तर्कों का अनुमान लगाएगा।

अनुभवी के लिए जावा 8 साक्षात्कार प्रश्न

12. लैम्ब्डा व्यंजक की मूल संरचना/वाक्यविन्यास क्या है?

```
FunctionalInterface fi = (स्ट्रिंग नाम) -> {System.out.println ("हैलो
" + नाम); वापसी "हैलो " + नाम; }
```

लैम्ब्डा अभिव्यक्ति को तीन अलग-अलग भागों में विभाजित किया जा सकता है:

1. तर्कों/परमों की सूची:

(स्ट्रिंग नाम)

पैराम्स की एक सूची () राउंड ब्रैकेट में पास की गई है। इसमें शून्य या अधिक पैरा हो सकते हैं।
पैरामीटर के प्रकार की घोषणा करना वैकल्पिक है और संदर्भ के लिए अनुमान लगाया जा सकता है।

2. तीर टोकन:

->

एरो टोकन को लैम्ब्डा एरो ऑपरेटर के रूप में जाना जाता है। इसका उपयोग मापदंडों को शरीर से अलग करने के लिए किया जाता है, या यह शरीर को तर्कों की सूची को इंगित करता है। 3.

अभिव्यक्ति / शरीर:

```
{ System.out.println ("हैलो " + नाम); वापसी "हैलो " +
नाम; }
```

एक निकाय में भाव या कथन हो सकते हैं। {} घुंघराले ब्रेसिज़ केवल तभी आवश्यक होते हैं जब एक से अधिक पंक्तियाँ हों। एक स्टेटमेंट में, रिटर्न टाइप स्टेटमेंट के रिटर्न टाइप के समान होता है। अन्य मामलों में, रिटर्न प्रकार का या तो रिटर्न कीवर्ड द्वारा अनुमान लगाया जाता है या यदि कुछ भी वापस नहीं किया जाता है तो शून्य हो जाता है।

13. लैम्ब्डा अभिव्यक्ति की विशेषताएं क्या हैं?

लैम्ब्डा अभिव्यक्तियों के रूप में परिभाषित विधियों की दो महत्वपूर्ण विशेषताएं नीचे दी गई हैं:

- लैम्ब्डा अभिव्यक्तियों को किसी अन्य विधि के पैरामीटर के रूप में पारित किया जा सकता है।
- लैम्ब्डा अभिव्यक्ति किसी भी वर्ग से संबंधित बिना स्टैंडअलोन हो सकती है।

14. एक प्रकार का इंटरफ़ेस क्या है?

टाइप इंटरफ़ेस जावा के पुराने संस्करणों में भी उपलब्ध है। इसका उपयोग कंपाइलर द्वारा कंपाइल समय पर मेथड इनवोकेशन और संबंधित डिक्लेरेशन को देखकर तर्क के प्रकार का अनुमान लगाने के लिए किया जाता है।

15. लैम्ब्डा अभिव्यक्तियों का उपयोग करने के प्रकार और सामान्य तरीके क्या हैं?

लैम्ब्डा अभिव्यक्ति का कोई विशिष्ट प्रकार नहीं होता है। एक लैम्ब्डा अभिव्यक्ति एक कार्यात्मक इंटरफ़ेस को सौंपे जाने के बाद टाइप प्राप्त करती है। उसी लैम्ब्डा अभिव्यक्ति को विभिन्न कार्यात्मक इंटरफ़ेस प्रकारों को सौंपा जा सकता है और एक अलग प्रकार हो सकता है।

उदाहरण के लिए अभिव्यक्ति s -> s.isEmpty() पर विचार करें:

```
विधेय <स्ट्रिंग> stringPredicate = s -> s.isEmpty ();
विधेय <सूची> सूची प्रेडिकेट = s -> s.isEmpty (); फ़ंक्शन <स्ट्रिंग, बूलियन> func = s ->
s.isEmpty ();
उपभोक्ता <स्ट्रिंग> stringConsumer = s -> s.isEmpty ();
```

अभिव्यक्ति का उपयोग करने के सामान्य तरीके

एक कार्यात्मक इंटरफ़ेस के लिए असाइनमेंट — > प्रेडिकेट<स्ट्रिंग> stringPredicate = s -> s.isEmpty (); एक पैरामीटर के रूप में पारित किया जा सकता है जिसमें एक कार्यात्मक प्रकार है -> stream.filter(s -> s.isEmpty ())

इसे किसी फ़ंक्शन से वापस करना -> वापसी s -> s.isEmpty ()
इसे एक कार्यात्मक प्रकार में कास्ट करना -> (Predicate<String>) s -> s.isEmpty()

16. Java 8 में, Method Reference क्या है?

विधि संदर्भ कार्यात्मक इंटरफ़ेस की विधि को संदर्भित करने का एक कॉम्पैक्ट तरीका है। इसका उपयोग किसी विधि को बिना बुलाए संदर्भित करने के लिए किया जाता है। :: (डबल कोलन) विधि संदर्भ का वर्णन करने के लिए प्रयोग किया जाता है। वाक्यविन्यास वर्ग :: विधि नाम है

उदाहरण के लिए:

```
पूर्णांक::parseInt(str) \\ विधि संदर्भ
str -> Integer.ParseInt (str); \\ समतुल्य लैम्ब्डा
```

17. String::ValueOf अभिव्यक्ति का क्या अर्थ है?

यह क्लास स्ट्रिंग के Valueof () विधि के लिए एक स्थिर विधि संदर्भ है। यह पारित तर्क के स्ट्रिंग प्रतिनिधित्व को वापस कर देगा।

18. एक वैकल्पिक वर्ग क्या है?

वैकल्पिक एक कंटेनर प्रकार है जिसमें मान हो सकता है या नहीं हो सकता है यानी शून्य (शून्य) या एक (शून्य-शून्य) मान। यह java.util पैकेज का हिस्सा है। isPresent() जैसी पूर्व-परिभाषित विधियां हैं, जो मान मौजूद होने पर सत्य लौटाती हैं या अन्यथा झूठी और विधि मिलती है (), जो मौजूद होने पर मान वापस कर देगी।

```
स्थिर वैकल्पिक <स्ट्रिंग> चेंजकेस (स्ट्रिंग शब्द) { अगर (नाम! = शून्य && शब्द। शुरू होता है
("ए")) { वापसी वैकल्पिक। (word.toUpperCase ()); } और { वापसी
    वैकल्पिक.ऑफ़नलेबल (शब्द); // someString अशक्त हो सकता है } }
```

Optional with **changeCase** object Empty Optional

InterviewBit

वैकल्पिक वर्ग

19. वैकल्पिक वर्ग का उपयोग करने के क्या लाभ हैं?

वैकल्पिक वर्ग का उपयोग करने के मुख्य लाभ नीचे दिए गए हैं:

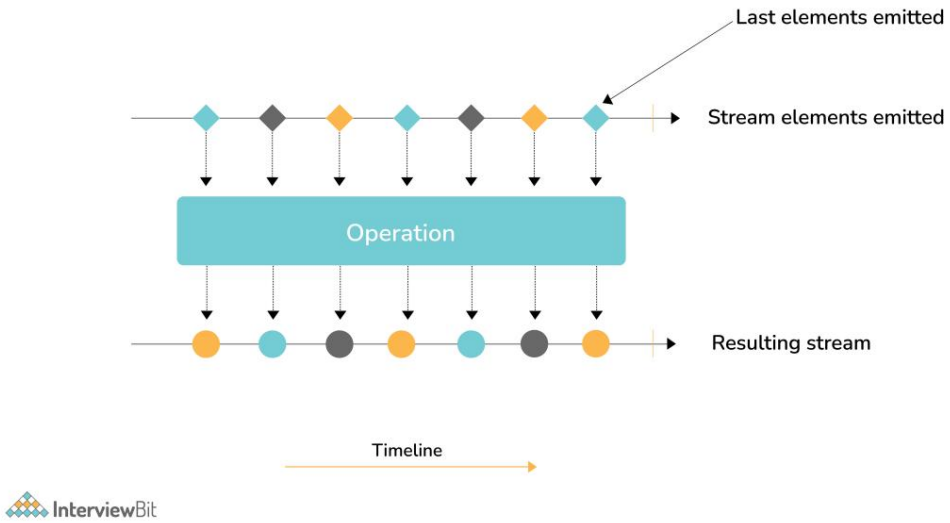
यह वैकल्पिक मानों को समाहित करता है, अर्थात, शून्य या शून्य-शून्य मान, जो अशक्त जाँच से बचने में मदद करता है, जिसके परिणामस्वरूप बेहतर, पठनीय और मजबूत कोड होता है। रन-टाइम NullPointerExceptions से बचने के लिए उपयोग किया जा सकता है।

20. जावा 8 स्ट्रीम क्या हैं?

घोषणात्मक तरीके से डेटा प्रोसेसिंग प्रश्नों को व्यक्त करने के लिए एक धारा एक अमूर्त है।

एक स्ट्रीम, जो डेटा ऑब्जेक्ट्स के अनुक्रम और उस डेटा पर संचालन की श्रृंखला का प्रतिनिधित्व करती है, एक डेटा पाइपलाइन है जो जावा I/O से संबंधित नहीं है स्ट्रीम स्थायी रूप से कोई डेटा नहीं रखती है।

मुख्य इंटरफ़ेस java.util.stream.Stream<T> है । यह फंक्शनल इंटरफेस को स्वीकार करता है ताकि लैम्ब्डा को पास किया जा सके। धाराएँ एक धाराप्रवाह इंटरफ़ेस या श्रृंखला का समर्थन करती हैं। नीचे मूल धारा समयरेखा संगमरमर आरेख है:

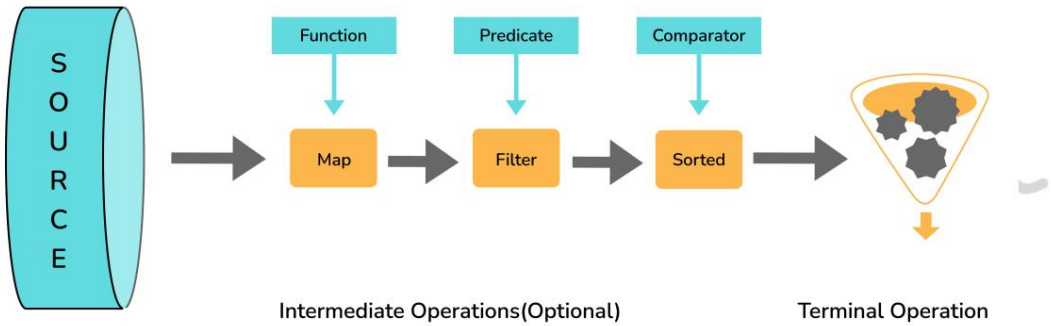


जावा 8 स्ट्रीम

21. धारा के मुख्य घटक क्या हैं?

धारा के घटक हैं:

- एक डेटा स्रोत
- डेटा स्रोत को संसाधित करने के लिए इंटरमीडिएट संचालन का सेट
- सिंगल टर्मिनल ऑपरेशन जो परिणाम उत्पन्न करता है



स्ट्रीम के घटक

22. डेटा ऑब्जेक्ट के स्रोत क्या हैं जो एक स्ट्रीम संसाधित कर सकता है?

एक स्ट्रीम निम्नलिखित डेटा को संसाधित कर सकती है:

- एक ऐरे का संग्रह।
- एक I/O चैनल या एक इनपुट डिवाइस।
- एक प्रतिक्रियाशील स्रोत (उदाहरण के लिए, सोशल मीडिया में टिप्पणियां या ट्वीट/री-ट्वीट)
- एक धारा जनरेटर समारोह या एक स्थिर कारखाना।

23. इंटरमीडिएट और टर्मिनल ऑपरेशन क्या हैं?

मध्यवर्ती संचालन:

- स्ट्रीम तत्वों को संसाधित करें।
- आमतौर पर एक धारा को दूसरी धारा में बदल देता है।
- आलसी हैं, यानी, एक टर्मिनल ऑपरेशन लागू होने तक निष्पादित नहीं किया जाता है।
- सभी स्रोत तत्वों का आंतरिक पुनरावृत्ति करता है।
- प्रसंस्करण पाइपलाइन में किसी भी संख्या में संचालन को जंजीर में बांधा जा सकता है।
- संचालन परिभाषित क्रम के अनुसार लागू होते हैं।
- इंटरमीडिएट ऑपरेशन ज्यादातर लैम्ब्डा फ़ंक्शन होते हैं।

टर्मिनल संचालन:

- स्ट्रीम पाइपलाइन को किक-स्टार्ट करता है। संसाधित
- स्ट्रीम डेटा एकत्र करने के लिए उपयोग किया जाता है।

```
int गिनती = Stream.of(1, 2, 3, 4, 5) .filter(i -> i <4) //
इंटरमीडिएट ऑपरेशन फ़िल्टर । गिनती (); // टर्मिनल ऑपरेशन गिनती
```

24. सबसे अधिक उपयोग किए जाने वाले इंटरमीडिएट ऑपरेशन क्या हैं?

Filter(Predicate<T>) - स्ट्रीम तत्वों के चयनात्मक प्रसंस्करण की अनुमति देता है। यह उन तत्वों को लौटाता है जो विधेय द्वारा आपूर्ति की गई स्थिति को संतुष्ट कर रहे हैं। map(Funtion<T, R>) - एक नया स्ट्रीम देता है, आपूर्ति

किए गए मैपर फ़ंक्शन को लागू करके प्रत्येक तत्व को रूपांतरित करता है। = सॉर्ट किया गया () - इनपुट तत्वों को सॉर्ट करता है और फिर उन्हें अगले चरण में भेजता है।

विशिष्ट () - केवल तत्वों को अगले चरण में पास करें, अभी तक पारित नहीं हुआ है। सीमा (लंबी अधिकतम आकार) -

स्ट्रीम आकार को अधिकतम करने के लिए सीमित करें।

छोड़ें (लंबी शुरुआत) - आरंभिक तत्वों को प्रारंभ तक छोड़ें।

पीक (उपभोक्ता) - स्ट्रीम में संशोधन किए बिना उपभोक्ता को लागू करें।

फ्लैटमैप (मैपर) - प्रत्येक तत्व को उसके घटक तत्वों की एक धारा में बदलें और सभी धाराओं को एक ही धारा में समतल करें।

25. स्टेटफुल इंटरमीडिएट ऑपरेशन क्या है? स्टेटफुल इंटरमीडिएट ऑपरेशंस के कुछ उदाहरण दें।

कुछ मध्यवर्ती कार्यों को पूरा करने के लिए, कुछ राज्य बनाए रखा जाना है, और इस तरह के मध्यवर्ती संचालन को स्टेटफुल इंटरमीडिएट ऑपरेशन कहा जाता है।
इस प्रकार के कार्यों का समानांतर निष्पादन जटिल है।

उदाहरण के लिए: क्रमबद्ध (), विशिष्ट (), सीमा (), छोड़ें () आदि।

डेटा तत्वों को पाइपलाइन में आगे के चरणों में भेजना तब तक रुक जाता है जब तक कि सभी डेटा सॉर्ट किए गए () और स्ट्रीम डेटा तत्वों को अस्थायी डेटा संरचनाओं में संग्रहीत नहीं किया जाता है।

26. टर्मिनल संचालन का सबसे सामान्य प्रकार क्या है?

- कलेक्ट () - स्ट्रीम अनुक्रम के सभी तत्वों से एकल परिणाम एकत्र करता है। कम करें () - स्ट्रीम अनुक्रम के सभी तत्वों से एकल
- परिणाम उत्पन्न करता है गिनती () - स्ट्रीम पर तत्वों की संख्या लौटाता है। मिनट () - स्ट्रीम से न्यूनतम तत्व लौटाता है। अधिकतम () -
  - स्ट्रीम से अधिकतम तत्व लौटाता है।
  -
  -
- खोज/क्वेरी संचालन anyMatch() ,
  - noneMatch() , allMatch() , ... - शॉर्ट सर्किट ऑपरेशन।
  - मैच की स्थिति के लिए इनपुट के रूप में एक विधेय लेता है।
  - जब और जब परिणाम निर्धारित किया जा सकता है, स्ट्रीम प्रोसेसिंग रोक दी जाएगी।
- प्रत्येक () के लिए पुनरावृत्त संचालन
  - - प्रत्येक स्ट्रीम तत्व के साथ कुछ करने के लिए उपयोगी। यह एक उपभोक्ता को स्वीकार करता है। forEachOrdered () - यह समानांतर धाराओं में क्रम बनाए रखने में मददगार है।
  -

27. फाइंडफर्स्ट () और फाइंडएनी () में क्या अंतर है?

फाइंडफर्स्ट ()	कोई भी खोजें()
स्ट्रीम में पहला तत्व लौटाता है	स्ट्रीम से कोई भी तत्व लौटाएं
प्रकृति में नियतात्मक	प्रकृति में गैर नियतात्मक

28. संग्रह, स्ट्रीम से किस प्रकार भिन्न हैं?

संग्रह स्ट्रीम के लिए स्रोत हैं। जावा 8 संग्रह एपीआई को संग्रह से Stream<T> लौटाने वाले डिफ़ॉल्ट तरीकों के साथ बढ़ाया गया है।

संग्रह	स्ट्रीम
डेटा संरचना में सभी डेटा तत्व होते हैं	कोई डेटा संग्रहीत नहीं है। मांग पर तत्वों की अनंत संख्या को संसाधित करने की क्षमता रखें
बाहरी यात्रा	आंतरिक पुनरावृत्ति
कई बार संसाधित किया जा सकता है	केवल एक बार ट्रैवर्स किया गया
तत्व हैं पहुँच में आसान	विशिष्ट तत्वों तक पहुँचने का कोई सीधा तरीका नहीं
डेटा स्टोर है	डेटा को संसाधित करने के लिए एक एपीआई है

29. जावा 8 में नई तिथि और समय एपीआई की विशेषता क्या है?

- अपरिवर्तनीय कक्षाएं और थ्रेड-सुरक्षित
- समय क्षेत्र समर्थन
- वस्तु निर्माण और अंकगणित के लिए धाराप्रवाह तरीके
- पहले के एपीआई के लिए I18N मुद्दे को संबोधित करता है
- लोकप्रिय जोडा-टाइम पैकेज से प्रभावित
- सभी पैकेज ISO-8601 कैलेंडर सिस्टम पर आधारित हैं

30. नए डेटा और समय के लिए महत्वपूर्ण पैकेज क्या हैं?

एपीआई?

- java.time
  - तिथियां
  - बार
  - झटपट
  - अवधि समय-
  - क्षेत्र अवधि
  -
- Java.time.format
- Java.time.temporal
- java.time.zone

31. उदाहरण के साथ समझाएं, लोकलडेट, लोकलटाइम, और लोकलडेटटाइम एपीआई।

स्थानीय तिथि

- कोई समय घटक के साथ दिनांक डिफ़ॉल्ट
- प्रारूप - yyyy-MM-dd (2020-02-20)
- लोकलडेट आज = लोकलडेट.अब (); // आज की तारीख देता है LocalDate aDate
- = LocalDate.of(2011, 12, 30); // (वर्ष, माह, तिथि)

स्थानीय समय

- नैनोसेकंड सटीक के साथ बिना तारीख वाला समय डिफ़ॉल्ट प्रारूप -
- hh:mm:ss:zzz (12:06:03.015) नैनोसेकंड वैकल्पिक है LocalTime now = LocalTime.now
- (); // अब समय देता है LocalTime aTime2 = LocalTime.of(18, 20, 30); // (घंटे, मिनट, सेकंड)
-

लोकलडेटटाइम

- दिनांक और समय दोनों डिफ़ॉल्ट प्रारूप रखता
- है - yyyy-MM-dd-HH-mm-ss.zzz (2020-02-20T12:06:03.015)
- LocalDateTime टाइमस्टैम्प = LocalDateTime.now (); // अब टाइमस्टैम्प देता है // (वर्ष, महीना, तिथि,
- घंटे, मिनट, सेकंड)
- LocalDateTime dt1 = LocalDateTime.of (2011, 12, 30, 18, 20, 30);

32. जावा में नैशॉर्न को परिभाषित करें 8

नैशॉर्न एक जावास्क्रिप्ट प्रोसेसिंग इंजन है जो जावा 8 के साथ बंडल किया गया है। यह ईसीएमए (यूरोपियन कंप्यूटर मैन्युफैक्चरर्स एसोसिएशन) सामान्यीकृत जावास्क्रिप्ट विनिर्देशों के साथ बेहतर अनुपालन और पुराने संस्करणों की तुलना में रन-टाइम पर बेहतर प्रदर्शन प्रदान करता है।

33. जावा 8 में जेजेएस का क्या उपयोग है?

जावा 8 के हिस्से के रूप में, जेजेएस एक कमांड-लाइन टूल है जो कंसोल में जावास्क्रिप्ट कोड को निष्पादित करने में मदद करता है। नीचे CLI कमांड का उदाहरण दिया गया है:

```
जावा>जेजेएस
jjs> प्रिंट ("नमस्ते, जावा 8 - मैं नया JJS हूँ!")
हैलो, जावा 8 - मैं नया जेजेएस हूं!
jjs> छोड़ें ()
>>
```

निष्कर्ष

कुल मिलाकर, जावा एक प्रचलित प्रोग्रामिंग भाषा है जो TIOBE और PYPL प्रोग्रामिंग भाषा रैंकिंग दोनों में लोकप्रियता में दूसरा स्थान हासिल करती है। दुनिया के अग्रणी टेक दिग्गज जैसे ट्विटर, लिंक्डइन, अमेज़ॅन, पेपाल, आदि अपने वेब ऐप और बैकएंड वेब सिस्टम बनाने के लिए जावा का उपयोग करते हैं। जावा भी Android ऐप्स विकसित करने के लिए उपयोग की जाने वाली प्राथमिक भाषाओं में से एक है; Google द्वारा समर्थित और प्रचारित एक ऑपरेटिंग सिस्टम।

आज तक, स्टैक ओवरफ्लो पर जावा के आसपास 1,751,661 प्रश्न हैं और गिटहब पर 123,776 जावा सार्वजनिक भंडार हैं और लगातार बढ़ रहे हैं। जावा 8 को सबसे स्थिर संस्करणों में से एक मानते हुए, इसमें करियर के अपार अवसर और गुंजाइश हैं। बस अवधारणाओं को समझें, उन्हें लागू करें और साक्षात्कार के लिए तैयार हो जाएं!

अतिरिक्त संसाधन

कोडिंग का अभ्यास करें

जावा ट्यूटोरियल

जावा साक्षात्कार के प्रश्न

अधिक साक्षात्कार के लिए लिंक प्रश्न

ग साक्षात्कार सवाल

पी एच पी सम्बंदित इन्टर्व्यू के सवाल

सी तीव्र साक्षात्कार प्रश्न

वेब एपीआई साक्षात्कार प्रश्न

हाइबरनेट साक्षात्कार प्रश्न

नोड जेएस साक्षात्कार प्रश्न

सीपीपी साक्षात्कार प्रश्न

उफ़ साक्षात्कार प्रश्न

देवोप्स साक्षात्कार प्रश्न

मशीन लर्निंग इंटरव्यू प्रश्न

डॉकर साक्षात्कार प्रश्न मैसकल साक्षात्कार प्रश्न

सीएसएस साक्षात्कार प्रश्न

लारवेल साक्षात्कार प्रश्न एएसपी नेट साक्षात्कार प्रश्न

Django साक्षात्कार प्रश्न डॉट नेट साक्षात्कार प्रश्न कुबेरनेट्स साक्षात्कार प्रश्न

ऑपरेटिंग सिस्टम साक्षात्कार प्रश्न

प्रतिक्रिया मूल निवासी साक्षात्कार प्रश्न

एडब्ल्यूएस साक्षात्कार प्रश्न

गिट साक्षात्कार प्रश्न

जावा 8 साक्षात्कार प्रश्न मोंगोडब साक्षात्कार प्रश्न

डीबीएमएस साक्षात्कार सवाल

स्प्रिंग बूट साक्षात्कार प्रश्न

पावर बीआई साक्षात्कार प्रश्न

Pl Sql साक्षात्कार प्रश्न

झांकी साक्षात्कार प्रश्न

लिनक्स साक्षात्कार प्रश्न

उत्तरदायी साक्षात्कार प्रश्न जावा साक्षात्कार प्रश्न

जेनकींस साक्षात्कार प्रश्न