JAVA INTERVIEW QUESTIONS

1. JDK - Java Development Kit

* It is the tool necessary to compile, document, and package Java programs.
* It contains JRE + development tools.

1. JRE - Java Runtime Environment

* JRE refers to a runtime environment in which Java bytecode can be executed.
* It’s an implementation of the JVM which physically exists.

1. JVM – Java Virtual Machine

* It is an abstract machine. It is a specification that provides a run-time environment in which Java bytecode can be executed.
* JVM follows three notations: Specification, Implementation, and Runtime Instance.

1. Explain public static void main(String args[]) in Java.

main() in Java is the entry point for any Java program. It is always written as public static void main(String[] args).

* public: Public is an access modifier used to specify who can access this method. Public means that this Method will be accessible by any Class.
* static: It is a keyword in java that identifies it as class-based. main() is made static in Java so that it can be accessed without creating an instance of a Class. In case, the main is not made static then the compiler will throw an error as main() is called by the JVM before any objects are made and only static methods can be directly invoked via the class.
* void: It is the return type of the method. Void defines the method which will not return any value.
* main: It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
* String args[]: It is the parameter passed to the main method.

1. Why Java is platform-independent?

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

1. Why Java is not 100% Object-oriented?

Java is not 100% Object-oriented because it makes use of eight primitive data types such as boolean, byte, char, int, float, double, long, and short which are not objects.

1. What are wrapper classes in Java?

Wrapper classes convert the Java primitives into the reference types (objects). Every primitive data type has a class dedicated to it. These are known as wrapper classes because they “wrap” the primitive data type into an object of that class. Refer to the below image which displays the different primitive types, wrapper classes, and constructor arguments.

1. What are constructors in Java?

In Java, a constructor refers to a block of code that is used to initialize an object. It must have the same name as that of the class. Also, it has no return type and it is automatically called when an object is created.

There are two types of constructors:

* Default Constructor: In Java, a default constructor is the one that does not take any inputs. Its main purpose is to initialize the instance variables with the default values. Also, it is majorly used for object creation.
* Parameterized Constructor: The parameterized constructor in Java, is the constructor which is capable of initializing the instance variables with the provided values. In other words, the constructors which take the arguments are called parameterized constructors.

1. What is a singleton class in Java and how can we make a class singleton?

Singleton class is a class whose only one instance can be created at any given time, in one JVM. A class can be made singleton by making its constructor private.

1. What is the difference between equals() and == in Java?

* Equals() method is defined in Object class in Java and used for checking the equality of two objects defined by business logic.
* “==” or equality operator in Java is a binary operator provided by Java programming language and used to compare primitives and objects.

1. When can you use the super keyword?

In Java, the super keyword is a reference variable that refers to an immediate parent class object.

The uses of the Java super Keyword are-

* To refer to an immediate parent class instance variable, use super.
* The keyword super can be used to call the method of an immediate parent class.
* Super() can be used to call the constructor of the immediate parent class.

1. What is the importance of reflection in Java?

Reflection is a runtime API for inspecting and changing the behavior of methods, classes, and interfaces. Java Reflection is a powerful tool that can be beneficial. Java Reflection allows you to analyze classes, interfaces, fields, and methods during runtime without knowing what they are called at compile time. Reflection can also be used to create new objects, call methods, and get/set field values. External, user-defined classes can be used by creating instances of extensibility objects with their fully qualified names. Debuggers can also use reflection to examine private members of classes.

1. How do not allow serialization of attributes of a class in Java?

The NonSerialized attribute can be used to prevent member variables from being serialized.

You should also make an object that potentially contains security-sensitive data non-serializable if possible. Apply the NonSerialized attribute to certain fields that store sensitive data if the object must be serialized. If you don’t exclude these fields from serialization, the data they store will be visible to any programs with serialization permission.

1. Can you call a constructor of a class inside another constructor?

Yes, we can call a constructor of a class inside another constructor. This is also called constructor chaining. Constructor chaining can be done in 2 ways-

* Within the same class: For constructors in the same class, this() keyword can be used.
* From the base class: The super() keyword is used to call the constructor from the base class.

1. Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.

An array generally contains elements of the primitive data types such as int, float, etc. In such cases, the array directly stores these elements at contiguous memory locations. While an ArrayList does not contain primitive data types. An ArrayList contains the reference of the objects at different memory locations instead of the object itself. That is why the objects are not stored at contiguous memory locations.

1. Why is it said that the length() method of the String class doesn’t return accurate results?

The length() method of the String class doesn’t return accurate results because

it simply takes into account the number of characters within the String. In other words, code points outside of the BMP (Basic Multilingual Plane), that is, code points having a value of U+10000 or above, will be ignored.

The reason for this is historical. One of Java’s original goals was to consider all text as Unicode; yet, Unicode did not define code points outside of the BMP at the time. It was too late to modify char by the time Unicode specified such code points.

1. Stack memory

* Stack memory is used only by one thread of execution.
* Stack memory can’t be accessed by other threads.
* Follows LIFO manner to free memory.
* Exists until the end of execution of the thread.
* Stack memory only contains local primitive and reference variables to objects in heap space.

1. Heap memory

* Heap memory is used by all the parts of the application.
* Objects stored in the heap are globally accessible.
* Memory management is based on the generation associated with each object.
* Heap memory lives from the start till the end of application execution.
* Whenever an object is created, it’s always stored in the Heap space.

1. What is a package in Java? List down various advantages of packages.

Packages in Java are the collection of related classes and interfaces which are bundled together. By using packages, developers can easily modularize the code and optimize its reuse. Also, the code within the packages can be imported by other classes and reused.

* Packages help in avoiding name clashes
* They provide easier access control on the code
* Packages can also contain hidden classes which are not visible to the outer classes and only used within the package
* Creates a proper hierarchical structure which makes it easier to locate the related classes

1. What is the JIT compiler in Java?

Just-In-Time compiler is a program that helps in converting the Java bytecode into instructions that are sent directly to the processor. By default, the JIT compiler is enabled in Java and is activated whenever a Java method is invoked. The JIT compiler then compiles the bytecode of the invoked method into native machine code, compiling it “just in time” to execute. Once the method has been compiled, the JVM summons the compiled code of that method directly rather than interpreting it. This is why it is often responsible for the performance optimization of Java applications at the run time.

1. Local variable

In Java, a local variable is typically used inside a method, constructor, or block and has only local scope. Thus, this variable can be used only within the scope of a block. The best benefit of having a local variable is that other methods in the class won’t be even aware of that variable.

1. Instance variable

An instance variable in Java is a variable that is bounded to its object itself. These variables are declared within a class, but outside a method. Every object of that class will create its copy of the variable while using it. Thus, any changes made to the variable won’t reflect in any other instances of that class and will be bound to that particular instance only.

1. What is the final keyword in Java?

Final is a special keyword in Java that is used as a non-access modifier.

1. Break

* Can be used in switch and loop (for, while, do-while) statements
* It causes the switch or loop statements to terminate the moment it is executed
* It terminates the innermost enclosing loop or switches immediately

1. Continue

* Can be only used with loop statements
* It doesn’t terminate the loop but causes the loop to jump to the next iteration
* A continue within a loop nested with a switch will cause the next loop iteration to execute

1. What is an infinite loop in Java?

An infinite loop is an instruction sequence in Java that loops endlessly when a functional exit isn’t met. This type of loop can be the result of a programming error or may also be a deliberate action based on the application behavior. An infinite loop will terminate automatically once the application exits.

1. What is Java String Pool?

Java String pool refers to a collection of Strings that are stored in heap memory. In this, whenever a new object is created, the String pool first checks whether the object is already present in the pool or not. If it is present, then the same reference is returned to the variable else new object will be created in the String pool and the respective reference will be returned.

1. Static

Is mainly used for memory management. Called without the class/ object instance. It uses early binding, the memory is fixed in the ram.

1. Non-static

Is everything by default. Called using an instance of class/ object. Uses late binding, the memory is not fixed in the ram.

1. Explain the term “Double Brace Initialisation” in Java?

Double Brace Initialization is a Java term that refers to the combination of two independent processes. There are two braces used in this. The first brace creates an anonymous inner class. The second brace is an initialization block. When these both are used together, it is known as Double Brace Initialisation. The inner class has a reference to the enclosing outer class, generally using the ‘this’ pointer. It is used to do both creation and initialization in a single statement. It is generally used to initialize collections. It reduces the code and also makes it more readable.

1. What is constructor chaining in Java?

In Java, constructor chaining is the process of calling one constructor from another concerning the current object. Constructor chaining is possible only through legacy where a subclass constructor is responsible for invoking the superclass constructor first. There could be any number of classes in the constructor chain. Constructor chaining can be achieved in two ways:

* Within the same class using this()
* From base class using super()

1. Why Java Strings are immutable?

In Java, string objects are immutable which simply means once the String object is created its state cannot be modified. Whenever you try to update the value of that object instead of updating the values of that particular object, Java creates a new string object. Java String objects are immutable as String objects are generally cached in the String pool. Since String literals are usually shared between multiple clients, action from one client might affect the rest. It enhances the security, caching, synchronization, and performance of the application.

1. What is the collection class in Java? List down its methods and interfaces.

In Java, the collection is a framework that acts as an architecture for storing and manipulating a group of objects. Using Collections you can perform various tasks like searching, sorting, insertion, manipulation, deletion, etc. Java collection framework includes the following:

* Interfaces
* Classes
* Methods

1. What is an association?

Association is a relationship where all object have their lifecycle and there is no owner.

Let’s take the example of Teacher and Student. Multiple students can associate with a single teacher and a single student can associate with multiple teachers but there is no ownership between the objects and both have their lifecycle. These relationships can be one to one, one to many, many to one, and many to many.

1. What do you mean by aggregation?

An aggregation is a specialized form of Association where all object has their lifecycle but there is ownership and a child object can not belong to another parent object.

Let’s take the example of Department and teacher. A single teacher can not belong to multiple departments, but if we delete the department teacher’s object will not destroy.

1. What is a composition in Java?

A composition is again a specialized form of Aggregation and we can call this a “death” relationship. It is a strong type of Aggregation. Child objects do not have their lifecycle and if the parent object deletes all child objects will also be deleted.

Let’s take again an example of a relationship between House and rooms. House can contain multiple rooms there is no independent life of room and any room can not belongs to two different houses if we delete the house room will automatically delete.

1. What is a marker interface?

In simpler terms, an empty interface is called the Marker interface. The most common examples of Marker interfaces in Java are Serializable, Cloneable.

1. What is object cloning in Java?

Object cloning in Java is the process of creating an exact copy of an object. It means the ability to create an object with a similar state as the original object. To achieve this, Java provides a method clone() to make use of this functionality. This method creates a new instance of the class of the current object and then initializes all its fields with the same contents as the corresponding fields. To object clone(), the marker interface java.lang.Cloneable must be implemented to avoid any runtime exceptions. One thing you must note is Object clone() is a protected method, thus you need to override it.

1. What is a servlet?

* Java Servlet is server-side technology to extend the capability of web servers by providing support for dynamic response and data persistence.
* The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing our own servlets.
* All servlets must implement the javax.servlet.Servlet interface, which defines servlet lifecycle methods. When implementing a generic service, we can extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet() and doPost(), for handling HTTP-specific services.
* Most of the time, web applications are accessed using the HTTP protocol, and that’s why we mostly extend HttpServlet class. The Servlet API hierarchy is shown below image.

1. What is a Request Dispatcher?

RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP, or another servlet in the same application. We can also use this to include the content of another resource in the response.

There are two methods defined in this interface:

* void forward()
* void include()

1. What is the life cycle of a servlet?

* Servlet is loaded
* Servlet is instantiated
* Servlet is initialized
* Service the request
* Servlet is destroyed

1. How do cookies work in Servlets?

* Cookies are text data sent by the server to the client and it gets saved at the client’s local machine.
* Servlet API provides cookies support through javax.servlet.http.Cookie class that implements Serializable and Cloneable interfaces.
* HttpServletRequest getCookies() method is provided to get the array of Cookies from the request, since there is no point in adding a Cookie to a request, there are no methods to set or add a cookie to a request.
* Similarly HttpServletResponse addCookie(Cookie c) method is provided to attach cookies in the response header, there are no getter methods for cookies.

1. What are the different methods of session management in servlets?

A session is a conversational state between client and server and it can consist of multiple requests and responses between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session-id) is passed between server and client in every request and response.

Some of the common ways of session management in servlets are:

* User Authentication
* HTML Hidden Field
* Cookies
* URL Rewriting
* Session Management API

1. POJO – Plain Old Java Object is an ordinary Java object that does not follow any of the major Java object models, conventions, or frameworks.
2. API – Application Programming Interface is software, that gives the possibility for 2 applications to communicate.
3. SOLID:

* Single responsibility – 1 method = 1 functionality
* Open – closed – open for extensions, closed for modifications
* Liskov substitution – subclass should be able to be replaced by the superclass
* Interface segregation – big interface made into smaller ones
* Dependency inversion – abstraction over concretization

1. Compile time – the time when the source code is converted to executable code.
2. Runtime – the time when the executable code is started running.

ВЪПРОСИ ЗА ИНТЕРВЮ С JAVA

1. JDK - Комплект за разработка на Java

        Това е инструментът, необходим за съставяне, документ и пакет Java програми.

        Той съдържа JRE + инструменти за разработка.

2. JRE - Java runtime среда

        JRE се отнася до среда по време на изпълнение, в която байтекодът на Java може да бъде изпълнен.

        Това е изпълнение на JVM, който физически съществува.

3. JVM – Виртуална машина Java

        Тя е абстрактна машина. Това е спецификация, която осигурява среда на изпълнение, в която java байтекод може да се изпълни.

        JVM следва три нотации: Спецификация, Внедряване и екземпляр по време на изпълнение.

4. Обяснете публична статична празнота main(Низа дъги[]) в Java.

main() в Java е входната точка за всяка java програма. Тя винаги е написана като публична статична празнота main(String[] арги).

        публичен: Публичен е модификатор за достъп, използван за указване кой може да получи достъп до този метод. Публично означава, че този Метод ще бъде достъпен по всеки Клас.

        статично: Това е ключова дума в java, която я идентифицира като базирана на клас. main() се прави статичен в Java, така че да може да бъде достъпен, без да се създава екземпляр на Клас. В случай, че основната не се прави статичен тогава компилаторът ще хвърли грешка като основен() се нарича от JVM, преди да бъдат направени каквито и да било обекти и само статични методи могат да бъдат директно извикани чрез класа.

        празнота: Той е типът връщане на метода. Void определя метода, който няма да върне никаква стойност.

        главно: Именно името на метода се търси от JVM като отправна точка за приложение само с определен подпис. Той е методът, при който се случва основното изпълнение.

        Струнни арги[]: Това е параметърът, предаден на основния метод.

5. Защо Java е независима от платформи?

Java се нарича платформа независима заради своите байт кодове, които могат да работят на всяка система независимо от основната си операционна система.

6. Защо Java не е 100% Обектно ориентирана?

Java не е 100% Обектно ориентиран, защото използва осем примитивни типа данни като булев, байт, чар, инт, плувка, двойно, дълго и кратко, които не са обекти.

7. Какво представляват класовете обвивка в Java?

Класовете обвивка преобразуват примитивните java в референтните типове (обекти). Всеки примитивния тип данни има клас, посветен на него. Те са известни като класове обвивка, защото те "увийте" примитивния тип данни в обект от този клас. Обърнете се към изображението по-долу, което показва различните примитивни типове, класовете на обвивката и аргументите на конструктора.

8. Какво представляват конструкторите в Java?

В Java конструктор се отнася до блок код, който се използва за инициализиране на обект. Сигурно има същото име като това на класа. Също така, той няма тип връщане и автоматично се извиква, когато се създава обект.

Има два вида конструктори:

        Конструктор по подразбиране: В Java конструктор по подразбиране е този, който не приема никакви входове. Основната му цел е да инициализира променливите на екземпляра със стойностите по подразбиране. Също така, той се използва основно за създаване на обекти.

        Параметризиран конструктор: Параметризираният конструктор в Java, е конструкторът, който е способен да инициализира екземплярните променливи с предоставените стойности. С други думи, конструкторите, които приемат аргументите, се наричат параметризирани конструктори.

9. Какво представлява един клас singleton в Java и как можем да направим клас singleton?

Singleton клас е клас, чийто единствен екземпляр може да бъде създаден по всяко време, в един JVM. Един клас може да се направи singleton, като си конструктор частен.

10. Каква е разликата между равни() и == в Java?

        Equals() метод се определя в обект клас в Java и се използва за проверка на равенството на два обекта, определени от бизнес логика.

                    "==" или оператор за равенство в Java е двоичен оператор, предоставен от java програмен език и използван за сравняване на примитивните и обектите.

11. Кога можете да използвате супер ключовата дума?

В Java супер ключовата дума е референтна променлива, която препраща към обект от непосредствен родителски клас.

Употребите на супер ключовата дума Java са-

        За да се обърнете към непосредствен екземпляр на родителски клас променлива, използвайте супер.

        Ключовата дума супер може да се използва за извикване на метода на непосредствен родителски клас.

        Супер() може да се използва за повикване на конструктора на непосредствения родителски клас.

12. Каква е важността на размисъла в Java?

Размисълът е aPI по време на изпълнение за проверка и промяна на поведението на методите, класовете и интерфейсите. Java Reflection е мощен инструмент, който може да бъде от полза. Java Reflection ви позволява да анализирате класове, интерфейси, полета и методи по време на изпълнение, без да знаете как се наричат по време на компилиране. Отражението може да се използва и за създаване на нови обекти, начини на повикване и получаване/задаване на стойности на полета. Външни, дефинирани от потребителя класове могат да се използват чрез създаване на екземпляри на обекти за разширяемост с техните напълно квалифицирани имена. Дебъгерите могат също да използват размисъл, за да изследват частни членове на класове.

13. Как да не се позволи сериализация на атрибути на клас в Java?

Атрибутът NonSerialized може да се използва, за да се предотврати сериализирането на променливите на членове.

Също така трябва да направите обект, който потенциално съдържа чувствителни към защитата данни, несериализиращи, ако е възможно. Приложете атрибута NonSerialized към определени полета, които съхраняват чувствителни данни, ако обектът трябва да бъде сериализиран. Ако не изключите тези полета от сериализацията, данните, които съхраняват, ще бъдат видими за всякакви програми с разрешение за сериализация.

14. Можете ли да се обадите на конструктор от клас вътре в друг конструктор?

Да, можем да се обадим на конструктор от клас вътре в друг конструктор. Това също се нарича конструктора верижно. Конструкторното верижно верига може да се направи по 2 начина-

        В рамките на същия клас: За конструктори от същия клас може да се използва тази() ключова дума.

        От базовия клас: Ключовата дума супер() се използва за извикване на конструктора от базовия клас.

15. Местата на прилежащата памет обикновено се използват за съхраняване на действителни стойности в масив, но не и в ArrayList. Обяснявам.

Масив по принцип съдържа елементи от примитивните типове данни като int, float и др. В такива случаи масивът директно съхранява тези елементи на места за прилежащата памет. Докато arrayList не съдържа примитивни типове данни. ArrayList съдържа препратката на обектите на различни места на паметта вместо на самия обект. Ето защо обектите не се съхраняват на места на прилежащата памет.

16. Защо се казва, че методът за дължина() на клас String не връща точни резултати?

Методът за дължина() на клас String не връща точни резултати, защото

тя просто отчита броя на знаците в рамките на Низ. С други думи, кодови точки извън BMP (Основна многоезична равнина), т.е. кодови точки, имащи стойност U+10000 или по-горе, ще бъдат игнорирани.

Причината за това е историческа. Една от първоначалните цели на Java беше да разгледа целия текст като Уникод; все още, Unicode не определи кодови точки извън BMP по това време. Беше твърде късно да се модифицира CHAR до момента, в който Unicode посочи такива кодови точки.

17. Стека памет

        Стека памет се използва само от една нишка на изпълнение.

        Стека памет не може да бъде достъпен от други нишки.

        Следва LIFO начин на свободна памет.

        Съществува до края на изпълнението на конеца.

        Стека памет съдържа само локални примитивни и референтни променливи на обекти в пространството на heap.

18. Heap памет

        Heap памет се използва от всички части на приложението.

        Обектите, съхранявани в чешмата, са достъпни в световен мащаб.

        Управлението на паметта се основава на поколението, свързано с всеки обект.

        Heap памет живее от самото начало до края на изпълнението на приложението.

        Винаги, когато се създава обект, той винаги се съхранява в пространството Heap.

19. Какво представлява пакетът в Java? Избройте различни предимства на пакетите.

Пакетите в Java са събирането на свързани класове и интерфейси, които са в пакет заедно. Като използват пакети, разработчиците могат лесно да модулират кода и да оптимизират повторното му използване. Също така, кодът в рамките на пакетите може да бъде импортиран от други класове и да се използвате повторно.

        Пакетите помагат при избягването на сблъсъци с имена

        Те осигуряват по-лесен контрол на достъпа върху кода

        Опаковките могат да съдържат и скрити класове, които не са видими за пределите на класовете и се използват само в рамките на опаковката

        Създава правилна йерархична структура, която улеснява откриването на свързаните класове

20. Какво представлява компилаторът на СЕР в Java?

Компилатор just-In-Time е програма, която помага при превръщането на байтекода на Java в инструкции, които се изпращат директно на процесора. По подразбиране компилаторът на СЕР е разрешен в Java и се активира винаги, когато се извиква метод на Java. Компилаторът на СЕР след това съставя байтекода на извиквания метод в родния машинен код, като го съставя "точно навреме", за да изпълни. След като методът е съставен, JVM по-скоро призлява съставяния код на този метод, вместо да го тълкува. Ето защо тя често е отговорна за оптимизацията на производителността на Java приложения в момента на изпълнение.

21. Локална променлива

В Java локална променлива обикновено се използва вътре метод, конструктор или блок и има само локален обхват. По този начин тази променлива може да се използва само в обхвата на блок. Най-добрата полза от наличието на локална променлива е, че други методи в класа няма дори да са наясно с тази променлива.

22. Променлива на екземпляра

Променлива на екземпляр в Java е променлива, която е обвързана със самия си обект. Тези променливи се декларират в рамките на даден клас, но извън даден метод. Всеки обект от този клас ще създаде своето копие на променливата, докато я използва. Така всички промени, направени в променливата, няма да отразяват в никакви други случаи от този клас и ще бъдат обвързани само с този конкретен екземпляр.

23. Каква е крайната ключова дума в Java?

Final е специална ключова дума в Java, която се използва като модификатор без достъп.

24. Прекъсване

        Може да се използва в превключвател и цикъл (за, докато, do-while) изявления

        Той кара комутаторът или изявленията за цикъл да прекратят момента, в който е изпълнена

        Той прекратява най-вътрешната ограждаща линия или превключва незабавно

25. Продължете

        Може да се използва само с изявления за цикъл

        Той не прекратява цикъла, а кара цикъла да прескочи до следващата ителация

        Продължаването в рамките на цикъл, вложен с превключвател, ще доведе до изпълнение на следващата ителация на цикъла

26. Какво е безкраен цикъл в Java?

Безкраен цикъл е последователност от инструкции в Java, която се примки безкрайно, когато функционален изход не е изпълнено. Този тип цикъл може да бъде резултат от програмна грешка или също може да бъде умишлено действие въз основа на поведението на приложението. Безкраен цикъл ще се прекрати автоматично, след като приложението излезе.

27. Какво е Струна басейн Java?

Java String басейн се отнася до колекция от Низове, които се съхраняват в heap памет. В това винаги, когато се създава нов обект, низът първо проверява дали обектът вече присъства в пула или не. Ако е налице, тогава същата препратка се връща към променливата друга нов обект ще бъде създаден в низа и съответната препратка ще бъде върната.

28. Статичен

Използва се главно за управление на паметта. Извикан без екземпляра на класа/ обекта. Той използва ранно свързване, паметта е фиксирана в овен.

29. Нестатични

Всичко по подразбиране ли е. Наречен с помощта на екземпляр на клас / обект. Използва късно свързване, паметта не е фиксирана в овен.

30. Обяснете термина "Двойна скоба Инициализация" в Java?

Double Brace Инициализация е java термин, който се отнася до комбинацията от два независими процеса. Има две скоби, използвани в това. Първата скоба създава анонимен вътрешен клас. Втората скоба е блок за инициализация. Когато тези и двете се използват заедно, той е известен като Double Brace Инициализация. Вътрешният клас има препратка към ограждащия външен клас, като по принцип се използва показалецът "този". Използва се, за да се направи както създаването, така и инициализацията в едно изявление. Обикновено се използва за инициализиране на колекции. Той намалява кода и също така го прави по-четим.

31. Какво представлява верижната верига на конструкторите в Java?

В Java, конструктор верижно е процесът на извикване на един конструктор от друг относно текущия обект. Верижното верига на конструктора е възможно само чрез наследство, когато подкласов конструктор е отговорен първо да се позове на конструктора на суперкласа. Може да има произволен брой класове във веригата на конструкторите. Конструктор верижно верига може да се постигне по два начина:

        В рамките на същия клас, използвайки това()

        От базов клас с помощта на супер()

32. Защо Java Strings са неизместими?

В Java струнните обекти са неизменни, което просто означава, след като обектът String бъде създаден състоянието му, не може да бъде модифициран. Винаги когато се опитвате да актуализирате стойността на този обект, вместо да актуализирате стойностите на този конкретен обект, Java създава нов низ обект. Обектите на Java String са неизмекливи, тъй като струнните обекти обикновено се кешират в низа. Тъй като струнните буквалности обикновено се споделят между няколко клиента, действието от един клиент може да повлияе на останалите. Той повишава защитата, кеширане, синхронизация и производителност на приложението.

33. Какъв е класът на събиране в Java? Избройте надолу неговите методи и интерфейси.

В Java колекцията е рамка, която действа като архитектура за съхранение и манипулиране на група обекти. Използване на Колекции можете да изпълнявате различни задачи като търсене, сортиране, вмъкване, манипулация, изтриване и др. Рамката за събиране на Java включва следното:

        Интерфейси

        Класове

        Методи

34. Какво представлява сдружението?

Асоциация е връзка, където всички обект имат своя жизнен цикъл и няма собственик.

Нека вземем примера на Учител и Ученик. Множество ученици могат да се свържат с един учител и един ученик може да се свърже с множество учители, но няма собственост между обектите и двете имат своя жизнен цикъл. Тези взаимоотношения могат да бъдат едно към едно, едно към много, много към едно и много към много.

35. Какво имате предвид под агрегация?

Агрегирането е специализирана форма на Асоциация, където целият обект има своя жизнен цикъл, но има собственост и дъщерен обект не може да принадлежи към друг родителски обект.

Да вземем примера на Отдел и учител. Един-единствен учител не може да принадлежи към множество отдели, но ако изтрием обекта на учителя по отдели няма да унищожи.

36. Какво е състав в Java?

Състав отново е специализирана форма на Агрегация и можем да наречем това "смърт" връзка. Той е силен тип Агрегация. Дъщерните обекти нямат жизнения си цикъл и ако родителският обект изтрие всички дъщерни обекти също ще бъдат изтрити.

Да вземем отново пример за връзка между Хаус и стаите. Къща може да съдържа множество стаи няма независим живот на стаята и всяка стая не може да принадлежи на две различни къщи, ако изтрием къщата стая автоматично ще изтрие.

37. Какво представлява интерфейсът на маркера?

С по-прости думи, празен интерфейс се нарича интерфейс marker. Най-често срещаните примери за Маркери интерфейси в Java са Serializable, Клонируеми.

38. Какво представлява клонирането на обекти в Java?

Клонирането на обекти в Java е процесът на създаване на точно копие на обект. Означава възможност за създаване на обект с подобно състояние като оригиналния обект. За да постигне това, Java осигурява метод клонинг() за използване на тази функционалност. Този метод създава нов екземпляр на класа на текущия обект и след това инициализира всичките му полета със същото съдържание като съответните полета. За да възразите срещу клонинг(), интерфейсът на маркера java.lang.Cloneable трябва да се внедри, за да се избегнат всякакви изключения по време на изпълнение. Едно нещо, което трябва да отбележите, е Обект клонинг() е защитен метод, като по този начин трябва да го презамести.

39. Какво е сервлет?

        Java Servlet е технология от страна на сървъра, за да разшири възможността на уеб сървърите, като предоставя поддръжка за динамичен отговор и упоритост на данните.

        Пакетите javax.servlet и javax.servlet.http предоставят интерфейси и класове за писане на наши собствени сервлети.

        Всички сервлети трябва да внедрят интерфейса javax.servlet.Servlet, който определя методите на жизнения цикъл на сервлет. При внедряването на генерична услуга можем да удължим класа GenericServlet, предоставен с API на Java Servlet. Класът HttpServlet предоставя методи, като doGet() и doPost(), за работа с http-специфични услуги.

        През по-голямата част от времето уеб приложенията са достъпни с помощта на HTTP протокола и затова най-вече удължаваме httpServlet класа. Йерархията на Servlet API е показана по-долу изображение.

40. Какво представлява диспечерът на заявки?

RequestDispatcher интерфейс се използва за препращане на заявката към друг ресурс, който може да бъде HTML, JSP или друга сервлет в същото приложение. Можем също така да използваме това, за да включим съдържанието на друг ресурс в отговора.

Има два метода, дефинирани в този интерфейс:

        празнота напред()

        празнота включват()

41. Какъв е жизнения цикъл на една сервичка?

        Сервлетът е зареден

        Сервлетът е моментиран

        Сервлетът се инициализира

        Обслужване на заявката

        Сервлетът е унищожен

42. Как работят бисквитките в Сервлетс?

        Бисквитките са текстови данни, изпратени от сървъра до клиента и той се записва в локалната машина на клиента.

        Servlet API предоставя поддръжка на бисквитки чрез javax.servlet.http.Cookie клас, който внедрява Serializable и Клонируеми интерфейси.

        HttpServletRequest getCookies() метод се предоставя, за да получите масива от Бисквитки от заявката, тъй като няма смисъл да добавяте "Бисквитка" към заявка, няма методи за задаване или добавяне на бисквитка към заявка.

        По същия начин HttpServletResponse addCookie(Cookie c) метод се предоставя за прикачване на бисквитки в заглавката на отговора, няма методи getter за бисквитки.

43. Какви са различните методи за управление на сесиите в сервлетите?

Сесията е разговорно състояние между клиент и сървър и тя може да се състои от множество заявки и отговори между клиент и сървър. Тъй като HTTP и Web Server и двете са без гражданство, единственият начин за поддържане на сесия е, когато някаква уникална информация за сесията (сесия-ид) се преминава между сървър и клиент във всяко искане и отговор.

Някои от общите начини за управление на сесиите в сервлетите са:

        Удостоверяване на потребителя

        HTML скрито поле

        Бисквитки

        Пренаписване на URL адреси

        API за управление на сесии

44. POJO – Plain Old Java Object е обикновен обект на Ява, който не следва нито един от основните модели обекти на Java, конвенции или рамки.

45. API – Application Programming Interface е софтуер, това дава възможност 2 приложения да комуникират.

46. ТВЪРДО:

        Единна отговорност – 1 метод = 1 функционалност

        Отворен – затворен – отворен за разширения, затворен за модификации

        Лисковската замяна – подкласът следва да може да бъде заменен със суперкласа

        Интерфейсна сегрегация – голям интерфейс, направен в по-малки

        Инверсия на зависимостта – абстракция над конкретизиране

47. Съставяне на време – времето, когато изходният код се преобразува в изпълним код.

48. Runtime – времето, когато изпълнимият код е стартиран.