**P # 1) ¿Qué es JAVA?**

**Responder:** Java es un lenguaje de programación de alto nivel y es independiente de la plataforma.

Java es una colección de objetos. Fue desarrollado por Sun Microsystems. Hay muchas aplicaciones, sitios web y juegos que se desarrollan con Java.

**P # 2) ¿Cuáles son las características de JAVA?**

**Respuesta: Las características de Java son las siguientes:**

* **Conceptos de POO**
  + Orientado a objetos
  + Herencia
  + Encapsulamiento
  + Polimorfismo
  + Abstracción
* **Plataforma independiente:** Un solo programa funciona en diferentes plataformas sin ninguna modificación.
* **Alto rendimiento:** JIT (compilador Just In Time) permite un alto rendimiento en Java. JIT convierte el código de bytes en lenguaje de máquina y luego JVM inicia la ejecución.
* **Multi-hilo:** Un flujo de ejecución se conoce como subproceso. JVM crea un hilo que se llama hilo principal. El usuario puede crear múltiples hilos extendiendo la clase de hilo o implementando la interfaz Runnable.

**P # 3) ¿Cómo permite Java un alto rendimiento?**

**Responder:** Java utiliza el compilador Just In Time para permitir un alto rendimiento. Se utiliza para convertir las instrucciones en códigos de bytes.

**P # 4) ¿Nombra los IDE de Java?**

**Responder:** Eclipse y NetBeans son los IDE de JAVA.

**P # 5) ¿Qué quieres decir con Constructor?**

**Respuesta: Constructor se puede explicar en detalle con puntos alistados:**

* Cuando se crea un nuevo objeto en un programa, se invoca un constructor correspondiente a la clase.
* El constructor es un método que tiene el mismo nombre que el nombre de la clase.
* Si un usuario no crea un constructor implícitamente, se creará un constructor predeterminado.
* El constructor puede estar sobrecargado.
* Si el usuario creó un constructor con un parámetro, entonces debería crear otro constructor explícitamente sin un parámetro.

**P # 6) ¿Qué se entiende por variable local y variable de instancia?**

**Responder:**

**Variables locales** se definen en el método y alcance de las variables que existen dentro del propio método.

**Instancia variable** se define dentro de la clase y fuera del método y el alcance de las variables existe en toda la clase.

**P # 7) ¿Qué es una clase?**

**Responder:** Todos los códigos de Java se definen en una clase. Tiene variables y métodos.

**Variables** son atributos que definen el estado de una clase.

**Métodos** son el lugar donde se debe realizar la lógica empresarial exacta. Contiene un conjunto de declaraciones (o) instrucciones para satisfacer el requisito particular.

**Ejemplo:**

|\_+\_|

**P # 8) ¿Qué es un objeto?**

**Responder:** Una instancia de una clase se llama objeto. El objeto tiene estado y comportamiento.

Siempre que la JVM lea la palabra clave 'new ()', creará una instancia de esa clase.

**P # 9) ¿Cuáles son los conceptos de POO?**

**Respuesta: Los conceptos de OOP incluyen:**

* Herencia
* Encapsulamiento
* Polimorfismo
* Abstracción
* Interfaz

***Lectura sugerida = >>***[***Principales preguntas de la entrevista de OOP***](https://spa.myservername.com/top-30-oops-interview-questions)

**P # 10) ¿Qué es la herencia?**

**Responder:** La herencia significa que una clase puede extenderse a otra clase. Para que los códigos se puedan reutilizar de una clase a otra. La clase existente se conoce como Superclase, mientras que la clase derivada se conoce como subclase.

La herencia solo se aplica al público y a los miembros protegidos. Los miembros privados no se pueden heredar.

**P # 11) ¿Qué es la encapsulación?**

**Respuesta: Propósito de la encapsulación:**

* Protege el código de otros.
* Mantenibilidad del código.

**Ejemplo:**

Declaramos 'a' como una variable entera y no debe ser negativa.

Si alguien cambia la variable exacta como ' ***a = -5”***entonces es malo.

**Para superar el problema, debemos seguir los pasos a continuación:**

* Podemos hacer que la variable sea privada o protegida.
* Utilice métodos de acceso público como set y get.

**El siguiente código muestra el captador y el definidor.**

Se pueden proporcionar condiciones al configurar la variable.

Para la encapsulación, necesitamos hacer que todas las variables de instancia sean privadas y crear setter y getter para esas variables. Lo que a su vez obligará a otros a llamar a los establecedores en lugar de acceder a los datos directamente.

**P # 12) ¿Qué es el polimorfismo?**

**Responder:** Polimorfismo significa muchas formas.

Un solo objeto puede referirse a la superclase o subclase dependiendo del tipo de referencia que se llama polimorfismo.

**Ejemplo:**

|\_+\_|

Usando el tipo de referencia de Manipulación podemos llamar al método 'add ()' de la clase Addition. Esta habilidad se conoce como polimorfismo. El polimorfismo es aplicable para **primordial** y no para **sobrecarga** .

**P # 13) ¿Qué se entiende por anulación de método?**

**Respuesta: La anulación del método ocurre si el método de subclase satisface las siguientes condiciones con el método de Superclase:**

* El nombre del método debe ser el mismo
* El argumento debe ser el mismo
* El tipo de retorno también debe ser el mismo

El beneficio clave de anular es que la subclase puede proporcionar información específica sobre ese tipo de subclase que la superclase.

**Ejemplo:**

**adición.add ()** El método llama al método add () en la subclase y no a la clase principal. Por lo tanto, anula el método Superclase y se conoce como Anulación de método.

**P # 14) ¿Qué se entiende por sobrecarga?**

**Responder:** La sobrecarga de métodos ocurre para diferentes clases o dentro de la misma clase.

**Para la sobrecarga de métodos, el método de subclase debe satisfacer las siguientes condiciones con el método de Superclase (o) métodos en la misma clase:**

* Mismo nombre de método
* Diferentes tipos de argumentos
* Puede haber diferentes tipos de devolución

**Ejemplo:**

|\_+\_|

Aquí, el método add () tiene diferentes parámetros en la clase Addition y está sobrecargado en la misma clase que con la superclase.

**Nota:** El polimorfismo no es aplicable a la sobrecarga de métodos.

**P # 15) ¿Qué se entiende por interfaz?**

**Responder:** No se pueden lograr varias herencias en java. Para superar este problema, se introduce el concepto de interfaz.

Una interfaz es una plantilla que solo tiene declaraciones de métodos y no la implementación del método.

**Ejemplo:**

|\_+\_|

* Todos los métodos de la interfaz son internos **vacío abstracto público** .
* Todas las variables en la interfaz son internamente **final estática pública** eso es constantes.
* Las clases pueden implementar la interfaz y no extender.
* La clase que implementa la interfaz debe proporcionar una implementación para todos los métodos declarados en la interfaz.

**P # 16) ¿Qué se entiende por clase abstracta?**

**Responder:** Podemos crear la clase Abstract usando la palabra clave 'Abstract' antes del nombre de la clase. Una clase abstracta puede tener tanto métodos 'abstractos' como métodos 'no abstractos' que son una clase concreta.

**Método abstracto:**

El método que solo tiene la declaración y no la implementación se llama método abstracto y tiene la palabra clave llamada “resumen”. Las declaraciones terminan con punto y coma.

**Ejemplo:**

|\_+\_|

* Una clase abstracta también puede tener un método no abstracto.
* La subclase concreta que amplía la clase abstracta debe proporcionar la implementación de métodos abstractos.

**Q # 17) Diferencia entre Array y Array List.**

**Responder:** **La diferencia entre Array y Array List se puede entender en la siguiente tabla:**

|  |  |
| --- | --- |
| **Formación** | **Lista de arreglo** |
| El tamaño debe darse en el momento de la declaración de la matriz.  Cadena [] nombre = nueva Cadena [2] | Puede que no se requiera el tamaño. Cambia el tamaño de forma dinámica.  ArrayList name = new ArrayList |
| Para poner un objeto en una matriz, necesitamos especificar el índice.  nombre [1] = 'libro' | No se requiere índice.  name.add ('libro') |
| La matriz no tiene parámetros de tipo | ArrayList en java 5.0 están parametrizados.  Por ejemplo: este corchete de ángulo es un parámetro de tipo que significa una lista de String. |

**P # 18) Diferencia entre String, String Builder y String Buffer.**

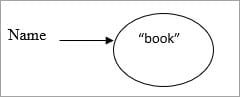
**Responder:**

**Cuerda:** Las variables de cadena se almacenan en un 'grupo de cadenas constante'. Una vez que la referencia de la cadena cambia el valor anterior que existe en el 'grupo de cadenas constante', no se puede borrar.

**Ejemplo:**

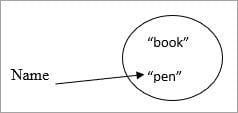
Nombre de cadena = 'libro';

**Pool de cuerdas constante**

.

Si el nombre-valor ha cambiado de 'libro' a 'pluma'.

**Pool de cuerdas constante**



Entonces, el valor más antiguo permanece en el grupo de cadenas constante.

**Búfer de cadena:**

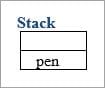
* Aquí los valores de cadena se almacenan en una pila. Si se cambian los valores, el nuevo valor reemplaza al anterior.
* El búfer de cadena está sincronizado, lo que es seguro para subprocesos.
* El rendimiento es más lento que el String Builder.

**Ejemplo:**

String Buffer name = ”libro”;



Una vez que el valor del nombre se ha cambiado a 'bolígrafo', el 'libro' se borra de la pila.



**Constructor de cadenas:**

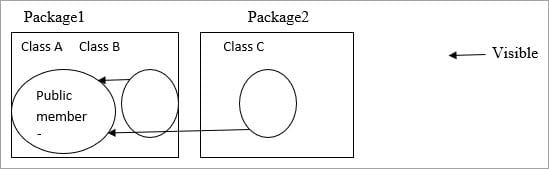
Esto es lo mismo que String Buffer, excepto por el String Builder que no está enhebrado de forma segura y no está sincronizado. Entonces, obviamente, el rendimiento es rápido.

**P # 19) Explique sobre los especificadores de acceso público y privado.**

**Responder:** Los métodos y las variables de instancia se conocen como miembros.

**Público:**

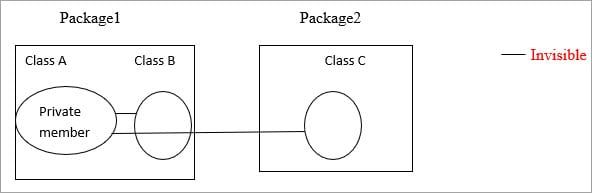
Los miembros públicos son visibles en el mismo paquete, así como en el paquete externo que es para otros paquetes.



Los miembros públicos de la Clase A son visibles para la Clase B (mismo paquete) y la Clase C (diferentes paquetes).

**Privado:**

Los miembros privados son visibles solo en la misma clase y no para las otras clases en el mismo paquete, así como las clases en los paquetes externos.



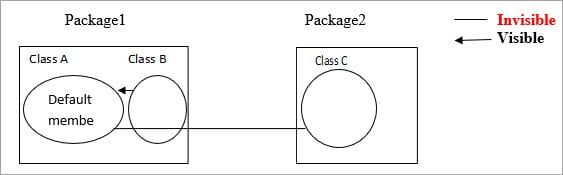
Los miembros privados de la clase A son visibles solo en esa clase. Es invisible para la clase B y la clase C.

**P # 20) Diferencia entre especificadores de acceso predeterminado y protegido.**

**Responder:**

[Ingeniero De Soporte Técnico Entrevista Preguntas Y Respuestas](https://spa.myservername.com/top-25-technical-support-interview-questions-with-answers)

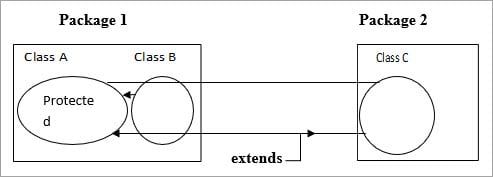
**Defecto:** Los métodos y variables declarados en una clase sin ningún especificador de acceso se denominan predeterminados.



Los miembros predeterminados en la Clase A son visibles para las otras clases que están dentro del paquete e invisibles para las clases que están fuera del paquete.

Por tanto, los miembros de Clase A son visibles para la Clase B e invisibles para la Clase C.

**Protegido:**

**.**

Protegido es lo mismo que Predeterminado, pero si una clase se extiende, es visible incluso si está fuera del paquete.

Los miembros de la clase A son visibles para la clase B porque están dentro del paquete. Para la Clase C es invisible, pero si la Clase C extiende la Clase A, los miembros son visibles para la Clase C incluso si está fuera del paquete.

**P # 21) Diferencia entre HashMap y HashTable.**

**Responder:** **La diferencia entre HashMap y HashTable se puede ver a continuación:**

|  |  |
| --- | --- |
| **HashMap** | **Tabla de picadillo** |
| Los métodos no están sincronizados | Los métodos clave están sincronizados |
| No seguridad para hilos | Seguridad del hilo |
| El iterador se usa para iterar los valores | El enumerador se usa para iterar los valores |
| Permite una clave nula y varios valores nulos | No permite nada que sea nulo |
| El rendimiento es alto que HashTable | El rendimiento es lento |

**P # 22) Diferencia entre HashSet y TreeSet.**

**Responder:** **La diferencia entre HashSet y TreeSet se puede ver a continuación:**

|  |  |
| --- | --- |
| **HashSet** | **TreeSet** |
| Los elementos insertados están en orden aleatorio | Mantiene los elementos en el orden ordenado |
| Puede almacenar objetos nulos | No se pudieron almacenar objetos nulos |
| El rendimiento es rápido | El rendimiento es lento |

**Q # 23) Diferencia entre clase abstracta e interfaz.**

**Respuesta: Las diferencias entre la clase abstracta y la interfaz son las siguientes:**

**Clase abstracta:**

* Las clases abstractas tienen un constructor predeterminado y se llama siempre que se crea una instancia de la subclase concreta.
* Contiene métodos abstractos y no abstractos.
* La clase que amplía la clase Abstract no debería requerir la implementación de todos los métodos, solo los métodos Abstract deben implementarse en la subclase concreta.
* La clase abstracta contiene variables de instancia.

**Interfaz:**

* No tiene ningún constructor y no se pudo crear una instancia.
* El método abstracto solo debe declararse.
* Las clases que implementan la interfaz deben proporcionar la implementación de todos los métodos.
* La interfaz contiene solo constantes.

**Q** **# 24) ¿Cuál es el significado de Colecciones en Java?**

**Responder:** La colección es un marco que está diseñado para almacenar los objetos y manipular el diseño para almacenar los objetos.

**Las colecciones se utilizan para realizar las siguientes operaciones:**

* buscando
* Clasificación
* Manipulación
* Inserción
* Supresión

Un grupo de objetos se conoce como colecciones. Todas las clases e interfaces para la recopilación están disponibles en el paquete util de Java.

**P # 25) ¿Cuáles son todas las clases e interfaces que están disponibles en las colecciones?**

**Responder:** **A continuación se muestran las clases e interfaces que están disponibles en las colecciones:**

**Interfaces:**

* Colección
* Lista
* Colocar
* Mapa
* Conjunto ordenado
* Mapa ordenado
* Cola

**Clases:**

* Liza:
* Lista de arreglo
* Vector
* Lista enlazada

**Conjuntos:**

* Conjunto de hash
* Conjunto de hash vinculado
* Árbol

**Mapas:**

* Mapa hash
* Tabla de picadillo
* TreeMap
* Mapa hash vinculado

**Cola:**

* Cola de prioridad

**P # 26) ¿Qué se entiende por Ordenado y Ordenado en colecciones?**

**Responder:**

**Ordenado:** Significa que los valores que se almacenan en una colección se basan en los valores que se agregan a la colección. Entonces podemos iterar los valores de la colección en un orden específico.

**Ordenado:** Los mecanismos de clasificación se pueden aplicar interna o externamente para que el grupo de objetos clasificados en una colección particular se base en las propiedades de los objetos.

**P # 27) Explique las diferentes listas disponibles en la colección.**

**Responder:** Los valores agregados a la lista se basan en la posición del índice y están ordenados por posición del índice. Se permiten duplicados.

**Los tipos de listas son:**

**a) Lista de matrices:**

* Iteración rápida y acceso aleatorio rápido.
* Es una colección ordenada (por índice) y no ordenada.
* Implementa la interfaz de acceso aleatorio.

**Producción:**

[Manzana, cereza, kiwi, plátano, cereza]

Desde la salida, Array List mantiene el orden de inserción y acepta los duplicados. Pero no está ordenado.

**b) Vector:**

Es lo mismo que Array List.

* Los métodos vectoriales están sincronizados.
* Seguridad del hilo.
* También implementa el acceso aleatorio.
* La seguridad de los subprocesos suele causar un impacto en el rendimiento.

**Producción:**

[cereza, manzana, plátano, kiwi, manzana]

Vector también mantiene el orden de inserción y acepta los duplicados.

**c) Lista vinculada:**

* Los elementos están doblemente vinculados entre sí.
* El rendimiento es más lento que el de la lista Array.
* Buena elección para inserción y eliminación.
* En Java 5.0, admite métodos de cola comunes peek (), Pool (), Offer (), etc.

**Producción:**

[plátano, cereza, manzana, kiwi, plátano]

Mantiene el orden de inserción y acepta los duplicados.

**Q # 28) Explique sobre Set y sus tipos en una colección.**

**Responder:** Set se preocupa por la singularidad. No permite duplicaciones. Aquí se utiliza el método 'igual ()' para determinar si dos objetos son idénticos o no.

**a) Conjunto de hash:**

* Desordenado y sin clasificar.
* Utiliza el código hash del objeto para insertar los valores.
* Use esto cuando el requisito sea 'no hay duplicados y no le importa el pedido'.

**Producción:**

[plátano, cereza, kiwi, manzana]

No sigue ningún orden de inserción. No se permiten duplicados.

**b) Conjunto de hash vinculado:**

* Una versión ordenada del conjunto de hash se conoce como Linked Hash Set.
* Mantiene una lista doblemente enlazada de todos los elementos.
* Úselo cuando se requiera una orden de iteración.

**Producción:**

[plátano, cereza, manzana, kiwi]

Mantiene el orden de inserción en el que se agregaron al conjunto. No se permiten duplicados.

**c) Conjunto de árboles:**

* Es una de las dos colecciones ordenadas.
* Utiliza la estructura de árbol 'Read-Black' y garantiza que los elementos estarán en orden ascendente.
* Podemos construir un conjunto de árboles con el constructor usando un comparador comparable (o).

**Producción:**

[manzana, plátano, cereza, kiwi]

TreeSet ordena los elementos en orden ascendente. Y no se permiten duplicados.

**P # 29) Explique sobre Map y sus tipos.**

**Respuesta: Mapa** se preocupa por el identificador único. Podemos asignar una clave única a un valor específico. Es un par clave / valor. Podemos buscar un valor, basándonos en la clave. Al igual que el conjunto, el mapa también utiliza el método 'equals ()' para determinar si dos claves son iguales o diferentes.

**El mapa es de los siguientes tipos:**

**a) Mapa hash:**

* Mapa desordenado y sin clasificar.
* Hashmap es una buena opción cuando no nos importa el orden.
* Permite una clave nula y varios valores nulos.

**Producción:**

{key2 = banana, key1 = cherry, key4 = kiwi, key3 = apple}

No se permiten claves duplicadas en Map.

No mantiene ningún orden de inserción y no está clasificado.

**b) Tabla hash:**

* Al igual que la clave vectorial, los métodos de la clase están sincronizados.
* Seguridad del hilo y, por tanto, ralentiza el rendimiento.
* No permite nada que sea nulo.

**Producción:**

{key2 = manzana, key1 = cherry, key4 = kiwi, key3 = banana}

No se permiten claves duplicadas.

**c)** **Mapa de hash vinculado:**

* Mantiene el orden de inserción.
* Más lento que el mapa Hash.
* Puedo esperar una iteración más rápida.

**Producción:**

{key2 = manzana, key1 = cherry, key4 = kiwi, key3 = banana}

No se permiten claves duplicadas.

[Cómo Comparar 2 Archivos En Unix](https://spa.myservername.com/how-compare-two-files-unix)

**d) TreeMap:**

* Mapa ordenado.
* Como Tree set, podemos construir un orden de clasificación con el constructor.

**Producción:**

{clave1 = cereza, clave2 = plátano, clave3 = manzana, clave4 = kiwi}

Está ordenado en orden ascendente según la clave. No se permiten claves duplicadas.

**P # 30) Explique la cola de prioridad.**

**Respuesta: Interfaz de cola**

**Cola de prioridad:** La clase de lista vinculada se ha mejorado para implementar la interfaz de cola. Las colas se pueden manejar con una lista vinculada. El propósito de una cola es 'Prioridad de entrada, Prioridad de salida'.

Por tanto, los elementos se ordenan de forma natural o según el comparador. El orden de los elementos representa su prioridad relativa.

**P # 31) ¿Qué se entiende por excepción?**

**Responder:** Una excepción es un problema que puede ocurrir durante el flujo normal de ejecución. Un método puede generar una excepción cuando algo suena en tiempo de ejecución. Si esa excepción no se pudo manejar, entonces la ejecución se termina antes de que complete la tarea.

Si manejamos la excepción, entonces el flujo normal continúa. Las excepciones son una subclase de java.lang.Exception.

**P # 32) ¿Cuáles son los tipos de excepciones?**

**Responder:** Hay dos tipos de excepciones. Se explican a continuación en detalle.

**a) Excepción marcada:**

El compilador comprueba estas excepciones en el momento de la compilación. Las clases que amplían la clase Throwable excepto la excepción de tiempo de ejecución y el error se denominan excepción marcada.

Las excepciones marcadas deben declarar la excepción usando la palabra clave throws (o) rodeada por el try / catch apropiado.

**Por ejemplo,** Excepción ClassNotFound

**b) Excepción sin marcar:**

El compilador no comprueba estas excepciones durante el tiempo de compilación. El compilador no obliga a manejar estas excepciones. **Incluye:**

* Excepción aritmética
* Excepción ArrayIndexOutOfBounds

**P # 33) ¿Cuáles son las diferentes formas de manejar las excepciones?**

**Responder:** **A continuación, se explican dos formas diferentes de manejar las excepciones:**

**a) Usando try / catch:**

El código de riesgo está rodeado por un bloque try. Si ocurre una excepción, entonces es capturada por el bloque catch que es seguido por el bloque try.

**Ejemplo:**

**b) Declarando la palabra clave throws:**

Al final del método, podemos declarar la excepción usando la palabra clave throws.

**P # 34) ¿Cuáles son las ventajas del manejo de excepciones?**

**Respuesta: Las ventajas son las siguientes:**

* El flujo normal de ejecución no terminará si se maneja una excepción
* Podemos identificar el problema utilizando la declaración de captura

**P # 35) ¿Cuáles son las palabras clave de manejo de excepciones en**

**Java?**

**Respuesta: A continuación se enumeran las dos palabras clave de manejo de excepciones:**

**un intento:**

Cuando un código de riesgo está rodeado por un bloque de prueba. Una excepción que ocurre en el bloque try es detectada por un bloque catch. El intento puede ser seguido por catch (o) finalmente (o) ambos. Pero cualquiera de los bloques es obligatorio.

**b) captura:**

A esto le sigue un bloque de prueba. Las excepciones se capturan aquí.

**c) finalmente:**

A esto le sigue el bloque try (o) el bloque catch. Este bloque se ejecuta independientemente de una excepción. Por lo tanto, aquí se proporcionan códigos de limpieza generales.

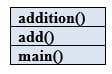
**P # 36) Explique sobre la propagación de excepciones.**

**Responder:** La excepción se lanza primero desde el método que está en la parte superior de la pila. Si no se detecta, aparece el método y se mueve al método anterior y así sucesivamente hasta que se obtienen.

Esto se denomina propagación de excepciones.

**Ejemplo:**

**En el ejemplo anterior, la pila se ve como se muestra a continuación:**

****

Si ocurre una excepción en el **adición()** el método no se captura, luego se mueve al método **agregar()** . Luego se mueve a la **principal()** método y luego detendrá el flujo de ejecución. Se llama Propagación de excepciones.

**P # 37) ¿Cuál es la palabra clave final en Java?**

**Responder:**

**Final variable:** Una vez que una variable se declara como final, el valor de la variable no se puede cambiar. Es como una constante.

**Ejemplo:**

int final = 12;

**Método final:** Una palabra clave final en un método, no se puede anular. Si un método está marcado como final, la subclase no puede anularlo.

**Clase final:** Si una clase se declara como final, entonces la clase no puede ser subclasificada. Ninguna clase puede extender la clase final.

**P # 38) ¿Qué es un hilo?**

**Responder:** En Java, el flujo de ejecución se denomina Thread. Cada programa java tiene al menos un hilo llamado hilo principal, el hilo principal es creado por JVM. El usuario puede definir sus propios hilos ampliando la clase Thread (o) implementando la interfaz Runnable. Los subprocesos se ejecutan al mismo tiempo.

**Q # 39) ¿Cómo se hace un hilo en Java?**

**Responder:** Hay dos formas disponibles para hacer un hilo.

**a) Ampliar la clase Thread:** Extender una clase Thread y anular el método de ejecución. El hilo está disponible en java.lang.thread.

La desventaja de usar una clase de hilo es que no podemos extender ninguna otra clase porque ya hemos extendido la clase de hilo. Podemos sobrecargar el método run () en nuestra clase.

**b) Implementar la interfaz Runnable:** Otra forma es implementando la interfaz ejecutable. Para eso, debemos proporcionar la implementación del método run () que se define en la interfaz.

**Q # 40) Explique sobre el método join ().**

**Responder:** El método Join () se usa para unir un hilo con el final del hilo que se está ejecutando actualmente.

Basado en el código anterior, el hilo principal ha comenzado la ejecución. Cuando llega al código ***t.start ()***luego 'thread t' inicia la propia pila para la ejecución. JVM cambia entre el hilo principal y el 'hilo t'.

Una vez que llega al código ***t.join ()***luego, 'thread t' solo se ejecuta y completa su tarea, luego solo el thread principal inicia la ejecución.

Es un método no estático. El método Join () tiene una versión sobrecargada. Por tanto, podemos mencionar la duración del tiempo en el método join () también '.s'.

**P # 41) ¿Qué hace el método de rendimiento de la clase Thread?**

**Responder:** Un método yield () mueve el subproceso que se está ejecutando actualmente a un estado ejecutable y permite que los otros subprocesos se ejecuten. Para que los subprocesos de igual prioridad tengan la oportunidad de ejecutarse. Es un método estático. No libera ningún bloqueo.

El método Yield () mueve el hilo de regreso al estado Ejecutable solamente, y no el hilo al bloque sleep (), wait () (o).

**Ejemplo:**

|\_+\_|

**P # 42) Explique sobre el método wait ().**

**Respuesta: espera ()** El método se usa para hacer que el hilo espere en el grupo de espera. Cuando se ejecuta el método wait () durante la ejecución de un hilo, inmediatamente el hilo abandona el bloqueo del objeto y va al grupo de espera. El método Wait () le dice al hilo que espere un tiempo determinado.

Luego, el hilo se activará después de que se llame al método notificar () (o) notificar a todos ().

Wait () y los otros métodos mencionados anteriormente no bloquean el objeto inmediatamente hasta que el subproceso que se está ejecutando actualmente completa el código sincronizado. Se utiliza principalmente en sincronización.

**Ejemplo:**

**Q # 43) Diferencia entre el método de notificación () y el método de notificación a todos () en Java.**

**Respuesta: Las diferencias entre el método de notificación () y el método de notificación a todos () se enumeran a continuación:**

|  |  |
| --- | --- |
| **notificar()** | **notificar a todos ()** |
| Este método se utiliza para enviar una señal para despertar un solo hilo en el grupo de espera. | Este método envía la señal para despertar todos los hilos en un carrete en espera. |

**P # 44) ¿Cómo detener un hilo en Java? ¿Explica sobre el método sleep () en un hilo?**

**Responder:** **Podemos detener un hilo usando los siguientes métodos de hilo:**

* Durmiendo
* Espera
* Obstruido

**Dormir:** El método Sleep () se utiliza para suspender el subproceso que se está ejecutando actualmente durante el tiempo especificado. Una vez que el hilo se activa, puede pasar al estado ejecutable. Por tanto, el método sleep () se utiliza para retrasar la ejecución durante algún tiempo.

Es un método estático.

**Ejemplo:**

**Hilo. Dormir (2000)**

Por lo que retrasa el hilo para dormir 2 milisegundos. El método Sleep () arroja una excepción ininterrumpida, por lo tanto, debemos rodear el bloque con try / catch.

**P # 45) ¿Cuándo usar la interfaz Runnable frente a la clase Thread en Java?**

**Responder:** Si necesitamos que nuestra clase extienda algunas otras clases además del hilo, entonces podemos ir con la interfaz ejecutable porque en Java solo podemos extender una clase.

Si no vamos a extender ninguna clase, podemos extender la clase de hilo.

**Q # 46) Diferencia entre el método start () y run () de la clase de hilo.**

**Responder:** El método Start () crea un nuevo hilo y el código dentro del método run () se ejecuta en el nuevo hilo. Si llamamos directamente al método run (), entonces no se crea un nuevo hilo y el hilo que se está ejecutando actualmente continuará ejecutando el método run ().

**P # 47) ¿Qué es el multihilo?**

**Responder:** Se ejecutan varios subprocesos simultáneamente. Cada hilo comienza su propia pila en función del flujo (o) prioridad de los hilos.

**Programa de ejemplo:**

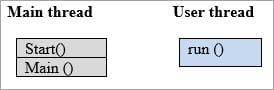
|\_+\_|

En la ejecución de la primera línea, JVM llama al método principal y la pila de subprocesos principal se ve como se muestra a continuación.

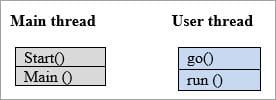


Una vez que llega la ejecución, **t.start ()** línea, se crea un nuevo hilo y también se crea la nueva pila para el hilo. Ahora JVM cambia al nuevo subproceso y el subproceso principal vuelve al estado ejecutable.

Las dos pilas se ven como se muestra a continuación.



Ahora, el hilo del usuario ejecutó el código dentro del método run ().



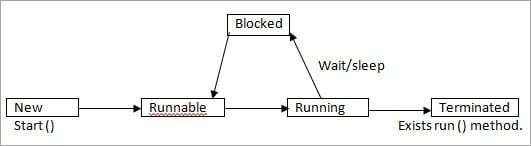
Una vez que se ha completado el método run (), JVM vuelve al hilo principal y el hilo del usuario ha completado la tarea y la pila ha desaparecido.

JVM cambia entre cada subproceso hasta que se completan ambos subprocesos. Esto se llama subprocesos múltiples.

**P # 48) Explique el ciclo de vida de los subprocesos en Java.**

**Responder:** **El hilo tiene los siguientes estados:**

* Nuevo
* Runnable
* Corriendo
* No ejecutable (bloqueado)
* Terminado



* **Nuevo:** En el estado Nuevo, se ha creado una instancia de Thread pero aún no se ha invocado el método start (). Ahora el hilo no se considera vivo.
* **Runnable** : El hilo está en estado ejecutable después de la invocación del método start (), pero antes de que se invoque el método run (). Pero un hilo también puede volver al estado ejecutable después de esperar / dormir. En este estado, el hilo se considera vivo.
* **Corriendo** : El hilo está en un estado de ejecución después de llamar al método run (). Ahora el hilo comienza la ejecución.
* **No ejecutable** (Bloqueado): el hilo está activo pero no es elegible para ejecutarse. No está en el estado ejecutable, pero también volverá al estado ejecutable después de algún tiempo. **Ejemplo:** espera, duerme, bloquea.
* **Terminado** : Una vez que se completa el método de ejecución, se termina. Ahora el hilo no está vivo.

**P # 49) ¿Qué es la sincronización?**

**Responder:** La sincronización hace que solo un hilo acceda a un bloque de código a la vez. Si varios subprocesos acceden al bloque de código, existe la posibilidad de obtener resultados inexactos al final. Para evitar este problema, podemos proporcionar sincronización para el bloque sensible de códigos.

La palabra clave sincronizada significa que un hilo necesita una clave para acceder al código sincronizado.

Los candados son por objetos. Cada objeto Java tiene un candado. Una cerradura tiene una sola llave. Un hilo puede acceder a un método sincronizado solo si el hilo puede obtener la clave de los objetos para bloquear.

Para ello, utilizamos la palabra clave 'Sincronizado'.