1. ¿Qué es un objeto y cuál es la diferencia entre un objeto y una clase?

Un objeto es un componente que se utiliza para crear la instancia de una clase, tiene métodos y propiedades.

Una clase es una plantilla para crear objetos.

1. ¿Qué es un constructor?

Un constructor es un elemento de una clase que se nombra igual a la clase correspondiente y que tiene por objetivo controlar cómo se inicializa una instancia de una determinada clase, en especial las variables de dicha clase.

1. ¿Cuáles son las diferencias entre String y **StringBuilder**?

StringBuilder son mutables, mientras que String es inmutable. Los objetos String se almacenan en el Constant String Pool que es un repositorio o almacén de cadenas, de valores de Strings. Esto se hace con el fin de que, si creamos otro String, con el mismo valor, no se cree un nuevo objeto sino que se use el mismo y se asigne una referencia al objeto ya creado. Los objetos StringBuilder se almacenan en el heap que es otro espacio de memoria usado en tiempo de ejecución para almacenar las instancias de clases, objetos y arrays.

1. ¿En términos de eficiencia, es mejor utilizar **String** o **StringBuilder** y por qué?

StringBuilder es la clase recomendada para la concatenación de String en la mayoría de los casos. Los métodos de StringBuilder no son sincronizados, por lo que tiene mejor rendimiento que String.

1. ¿Cuáles son las diferencias entre un método no estático y un método estático?

|  |  |
| --- | --- |
| Un método estático es un método que pertenece a una clase, pero no pertenece a una instancia de esa clase y este método se puede llamar sin la instancia u objeto de esa clase. | Todos los métodos en java tienen como valor predeterminado un método no estático sin una palabra clave estática que lo precede. Los métodos no estáticos pueden acceder a cualquier método y variable estáticos también, sin utilizar el objeto de la clase. |
| El método estático utiliza tiempo de compilación o enlace anticipado. | El método no estático usa tiempo de ejecución o enlace dinámico. |
| El método estático no se puede anular debido al enlace anticipado. | El método no estático se puede anular debido al enlace en tiempo de ejecución. |
| En el método estático, el método solo puede acceder a miembros de datos y métodos estáticos de otra clase o la misma clase, pero no puede acceder a métodos y variables no estáticos. | En el método no estático, el método puede acceder a miembros de datos y métodos estáticos, así como a miembros y métodos no estáticos de otra clase o la misma clase. |

Al igual que los métodos estáticos, una variable estática pertenece a la clase misma y una variable no estática pertenece a cada instancia de una clase. Por lo tanto, el valor de una variable estática sigue siendo el mismo para cada instancia de la clase, pero no se puede decir lo mismo de la variable no estática.

No podemos acceder a variables no estáticas dentro de un método estático sin una instancia de su clase. Un método estático solo puede acceder a variables estáticas, mientras que un método no estático puede acceder tanto a variables estáticas como no estáticas.

Las variables estáticas reducen la huella de memoria del programa. Esto se debe a que la memoria se asigna solo una vez para una variable estática durante el tiempo de carga de la clase, mientras que, para una variable no estática, la memoria se asigna cada vez que se crea una instancia de la clase.

Las variables estáticas generalmente se declaran finales en Java. Esto asegura que el valor nunca se cambie después de su inicialización. Esto es muy útil cuando necesitamos compartir una sola copia de la variable entre todas las instancias de la clase.

1. ¿Qué es el bloque static y para que nos sirve?

Su tarea principal es la de inicializar las variables estáticas, por ello son conocidos como inicializadores estáticos, en el ejemplo se inicializa la variable "numero", pero se preguntarán cual sería la diferencia de inicializarla en un método estático o directamente cuando se declara,

En comparación con la inicialización dentro de un método estático podemos mencionar que el bloque estático solo se ejecuta una sola vez cada que la clase a la que pertenece es cargada por un classloader, y el método estático se puede ejecutar cuantas veces se desee, es decir el número de veces que se invoque.

Por lo tanto, el uso de bloques estáticos se adecua más para la inicialización de constantes pues solo se ejecuta una sola ocasión (yo antes lo entendía como una especie de constructor para la parte estática del PermGen, aunque no es lo mismo, ni creo que sea correcto llamarlo así), lo cual no impide que también se usen bloques estáticos para dar un valor inicial a una variable que pueda cambiar durante la ejecución del programa.

1. ¿Qué es encapsulamiento y como implementarlo en una clase?

El término encapsulamiento en Java, consiste en ocultar atributos de un objeto de manera que solo se pueda cambiar mediante operaciones definidas en ese objeto. Está estrechamente relacionado con la visibilidad.

Para definir la visibilidad en Java, se dispone de palabras reservadas:

* public: nos indica que es accesible desde cualquier clase [interface].
* private: solo es accesible desde a clase actual.
* protected: accesible desde la clase actual, sus descendientes o el paquete del que forma parte.
* sin ninguna palabra: accesible desde cualquier clase del paquete.

Los atributos en Java, deben tener visibilidad private. Entonces, ¿cómo accedemos a los atributos? La respuesta está en los métodos Getters y Setters. A continuación se muestra un ejemplo de código de estos métodos.

class Punto {

private double puntox, puntoy;

public double getPuntox() {

return puntox;

}

public void setPuntox (double puntox) {

this.puntox = puntox;

}