Guía de Estudio: Clase 04

1. ¿Qué significa **sobrecargar** un método o constructor?
2. ¿Qué debe cambiar para que la sobrecarga de un método o constructor sea válida?
3. ¿La sobrecarga se resuelve en tiempo de ejecución o en tiempo de compilación? ¿Cómo se distingue a qué sobrecarga llamar?
4. ¿Se tiene en cuenta el nombre o identificador de los parámetros de entrada para una sobrecarga?
5. ¿Se tiene en cuenta el modificador de visibilidad para una sobrecarga?

Sobrecarga de Métodos:

6. ¿Los métodos pueden tener el mismo nombre que otros elementos de una misma clase? (atributos, propiedades, etc).
7. Mencione dos razones por las cuales se sobrecargan los métodos.
8. ¿Los métodos estáticos pueden ser sobrecargados?
9. ¿Agregar el modificador “static” sin cambiar los parámetros de entrada es una sobrecarga válida?
10. ¿Agregar un modificador “out” o “ref” en la firma del método sin cambiar nada más es una sobrecarga válida?
11. ¿Cambiar el tipo de retorno sin cambiar los parámetros de entrada es una sobrecarga válida?
12. Si tenemos distintas sobrecargas de un método, ¿cómo podemos reutilizar código?

Sobrecarga de Constructores:

13. ¿Se pueden sobrecargar los constructores estáticos?

14. ¿Se puede llamar a un constructor estático con el **operador “this()”**?
15. ¿Se puede llamar a constructores de otras clases con el operador “this()”?
16. ¿Se puede sobrecargar un constructor privado?

Sobrecarga de Operadores:

17. ¿Qué es un **operador**? ¿En qué se diferencian un operador **unario** y un operador **binario**? De un ejemplo de cada uno.
18. ¿Qué varía en la sintaxis de la sobrecarga de operadores unarios y binarios?
19. ¿Se pueden sobrecargar los operadores de operación y asignación (+=, -=, *=, /=)? ¿Por qué?
20. ¿Cuál es la diferencia entre un **operador de conversión implícito** y uno **explícito**? (En finalidad, declaración y aplicación)
21. Los operadores de casteo “(T)x” no se pueden sobrecargar. ¿Cuál es la alternativa?
22. ¿Cuál es la diferencia entre **castear** (casting), **convertir** (converting) y **parsear** (parsing)?