1. ¿Qué distingue a los lenguajes declarativos de los lenguajes imperativos?

- **Declarativos:** Se enfocan en *qué* se debe hacer, especificando el resultado deseado sin describir cómo alcanzarlo. Ejemplo: SQL, Prolog.
- **Imperativos:** Se enfocan en *cómo* se debe hacer, describiendo una secuencia de pasos para alcanzar el resultado. Ejemplo: C, Python.

---

2. Nombre un lenguaje de programación que sea interpretado y compilado y describa su comportamiento.

- **Java:**
    - El código fuente se compila en bytecode (usando el compilador javac).
    - El bytecode se interpreta o ejecuta mediante la Máquina Virtual de Java (JVM).
    - Esto permite portabilidad y optimización en tiempo de ejecución.

---

3. ¿Qué se entiende por abstracción?

- Es la capacidad de ocultar los detalles de implementación y exponer solo las funcionalidades esenciales, permitiendo trabajar con representaciones más simples y generales.

---

4. ¿Qué es un Registro de Activación (Stack Frame)? ¿Cuál es su estructura?

- Es una estructura que representa la información de una función o método durante su ejecución en la pila de llamadas.
- **Estructura típica:**
    - Valor de retorno.
    - Parámetros de entrada.
    - Variables locales.
    - Dirección de retorno (dónde reanudar tras la ejecución).

---

5. ¿Qué se entiende por programación estructurada o modular?

- Es un paradigma que fomenta la división del programa en bloques o módulos más pequeños (funciones, procedimientos) para mejorar legibilidad, reutilización y mantenibilidad.

---

6. ¿Qué se entiende por encapsulación?

- Es el principio de agrupar datos (atributos) y comportamientos (métodos) en una entidad (clase) y restringir el acceso directo mediante modificadores de visibilidad (privado, protegido, público).

---

7. ¿Cómo se definen las condiciones de Bernstein?

- Son reglas que determinan si dos o más tareas pueden ejecutarse en paralelo sin conflictos, evaluando dependencias de entrada, salida y antidependencias.

---

8. ¿Cuáles son las principales características de la programación paralela?

- **Características:**
    - Ejecución simultánea de tareas en múltiples procesadores.
    - Coordinación y sincronización de tareas.
    - Escalabilidad y optimización del tiempo de ejecución.

---

9. ¿Cómo se define una closure?

- Es una función que:
    - Puede capturar variables del ámbito donde fue definida.
    - Mantiene estas variables disponibles incluso después de que el ámbito haya finalizado.

---

10. ¿Qué es un contexto (binding)?

- Es la asociación de variables, métodos y su entorno de ejecución en un momento dado. Ruby, por ejemplo, permite capturar el contexto usando el objeto binding.

---

11. ¿Qué son las funciones de orden superior (high order functions)?

- Son funciones que:
    - Pueden recibir otras funciones como argumentos.
    - O devolver otras funciones como resultado.

---

12. ¿Cuál es la diferencia entre un tipo y clase?

- **Tipo:** Describe el conjunto de valores y operaciones permitidas (concepto más abstracto).
- **Clase:** Es una implementación concreta de un tipo en un lenguaje orientado a objetos.

---

13. Diferencia entre proceso y thread

- **Proceso:**
  - Instancia independiente de un programa en ejecución.
  - Tiene su propio espacio de memoria.
- **Thread:**
  - Subunidad de un proceso.
  - Comparte memoria y recursos del proceso padre.

---

14. Ventajas y desventajas de un compilador frente a un intérprete

- **Ventajas:**
  - Mayor velocidad de ejecución (el código ya está traducido a lenguaje máquina).
  - Detección de errores antes de la ejecución.
- **Desventajas:**
  - Tiempo inicial de compilación.
  - Menos flexibilidad para cambios en tiempo de ejecución.

---

15. ¿Qué criterio usa la página TIOBE para hacer el ranking?

- TIOBE clasifica los lenguajes según su **popularidad**, utilizando métricas como:
  - Búsquedas en motores como Google, Bing, Yahoo.
  - Frecuencia de menciones en sitios web, foros y tutoriales.
  - Cantidad de desarrolladores usando el lenguaje.

---

16. Algoritmo de búsqueda de métodos en Ruby:

```
module M
  def self.my_method; 'M#my_method()'; end
end
class C
  include M
end
```

```
class D < C ; end

D.new.my_method()
```

1. Busca en la eigenclass del objeto creado. (No lo encuentra).

2. Busca en los métodos de instancia de la clase D (No lo encuentra).

3. Busca en los módulos incluidos de la clase D (No lo encuentra).

4. Busca en los métodos de instancia de la superclase que es C (No lo encuentra).

5. Busca en los módulos incluidos de la clase C, en concreto en M. No lo encuentra porque el self.my_method esta definido en la eigenclass del modulo.

6. Busca en los métodos de instancia y módulos incluidos de Object, no lo encuentra.

7. Hace lo mismo con BsicObject, no lo encuentra.

8. Se llama a 'method_missing' al no encontrarse el método.