UDS

Mi Universidad

NOMBRE DEL ALUMNO: MARLONG URIEL RAMOS

NOMBRE DEL PROFESOR: ANDRES ALEJANDRO REYES MOLINA

MATERIA: PROGRAMACION LOGICA

CUATRIMESTRE: 8

TIPO DE TRABAJO: SUPER NOTA

3.7 IMPLEMENTACIÓN	3.8 PEREZA Y AFÁN CONTROLAR EL ENTUSIASMO EN LENGUAJES PEREZOSOS	3.9 SIMULAR LA PEREZA EN IDIOMAS ÁVIDOS	3.10 LA ESTRATEGIA DE EVALUACIÓN PEREZOSA.	3.11 TÉCNICAS DE PROGRAMACIÓN FUNCIONAL PEREZOSA.
Algunos lenguajes de programación retrasan la evaluación de expresiones de forma predeterminada, y otros proporcionan funciones o sintaxis especial para retrasar la evaluación. En Miranda y Haskell , la evaluación de los argumentos de función se retrasa de forma predeterminada. En muchos otros idiomas, la evaluación puede ser retrasado mediante la suspensión de forma explícita el cálculo utilizando la sintaxis especial (como ocurre con el esquema de " delay" y " force" y OCaml 's ' lazy' y " Lazy.force") o, más generalmente, envolviendo la expresión en un golpe seco .	En los lenguajes de programación perezosos como Haskell, aunque el valor predeterminado es evaluar las expresiones sólo cuando se exigen, en algunos casos es posible hacer que el código sea más ansioso o, por el contrario, hacerlo más perezoso nuevamente después de que se haya hecho más ansioso. Esto se puede hacer codificando explícitamente algo que fuerce la evaluación (lo que puede hacer que el código sea más ansioso) o evitando dicho código (lo que puede hacer que el código sea más perezoso). La evaluación estricta suele implicar entusiasmo, pero son conceptos técnicamente diferentes.	Java En Java , la evaluación diferida se puede realizar mediante el uso de objetos que tienen un método para evaluarlos cuando se necesita el valor. El cuerpo de este método debe contener el código necesario para realizar esta evaluación. Desde la introducción de expresiones lambda en Java SE8, Java ha admitido una notación compacta para esto. El siguiente ejemplo de interfaz genérica proporciona un marco para la evaluación diferida: interfaz Lazy < T > { T eval () } La Lazyinterfaz con su eval()método es equivalente a la Supplierinterfaz con su get()método en la java.util.functionbiblioteca Cada clase que implementa la Lazyinterfaz debe proporcionar un evalmétodo, y las instancias de la clase pueden tener los valores que el método necesite para realizar una evaluación perezosa. Por ejemplo, considere el siguiente código para calcular e imprimir perezosamente 2: Perezoso < Entero > a = () -> 1 ; for (int i = 1 ; i <= 10 ; i ++) { final Lazy < Integer > b = a ; a = () -> b . eval () + b . eval (); } Sistema . fuera . println ("a =" + a . eval ());	Programar en funcional es pensar en funciones. Cualquier algoritmo puede - y debe bajo esta perspectiva - pensarse como un encadenamiento de expresiones funcionales. Sin embargo, la programación funcional no es sólo eso. Sobre todo, y ante todo, cualquier lenguaje de programación que se precie dentro de este paradigma se caracteriza por un comportamiento específico acerca de cómo evalúa estas construcciones. Entender bien este hecho es muy importante de cara a controlar el paradigma ya que como desarrolladores es importante saber qué espera nuestro lenguaje de nosotros y de nuestro código	La evaluación perezosa puede también reducir el consumo de memoria de una aplicación, ya que los valores se crean solo cua ndo se necesitan. Sin embargo, es difícil de combinar con las operaciones típicas de programación imperativa, como el manejo de excepciones o las operaciones de entrada/salida, porque el orden de las operaciones puede quedar indeterminado. Además, la evaluación perezosa puede conducir a fragmentar la memoria. Lo contrario de la evaluación perezosa sería la evaluación acaparadora, o evaluación estricta, que es el modo de evaluación por defecto en la mayoría de los lenguajes de programación.

https://plataformaeducativauds.com.mx/assets/docs/libro/ISC/ef1be5c3a038bcf8d038423a2807a36e-LC-ISC802%20PROGRAMACION%20%20LOGICA.pdf