

Paradigmas de programación.

Guía de estudio para el examen ETS.

Parte I. Conceptos generales.

1. Define los siguientes conceptos:

- Lenguaje de programación.
- Paradigma de programación.

2. Menciona las características de los siguientes paradigmas de programación:

- Paradigma declarativo.
- Paradigma imperativo.

3. Completa la siguiente tabla:

Lenguaje de programación	Tipo de paradigma que usa	Dominio(s) de aplicación
C		
C++		
Java		
Python		
ML		
Haskell		
Prolog		
JavaScript		
SQL		
Basic		
Fortran		
Pascal		
Cobol		
Algol		
Perl		

4. Define cada uno de los siguientes conceptos:

- Compilador.
- Intérprete.
- Máquina virtual.

5. Describe cada una de las etapas del compilador del lenguaje de programación C.

6. Describe cada una de las etapas del proceso de compilación/interpretación del lenguaje de programación Java.

Parte II. Programación funcional.

Definir los siguientes conceptos:

- Características de la programación funcional.
- Expresiones lambda
- Funciones de primer orden (o funciones de orden superior).
- Evaluación estricta (*eager*) y evaluación no estricta (*lazy*)
- Funciones polimórficas.

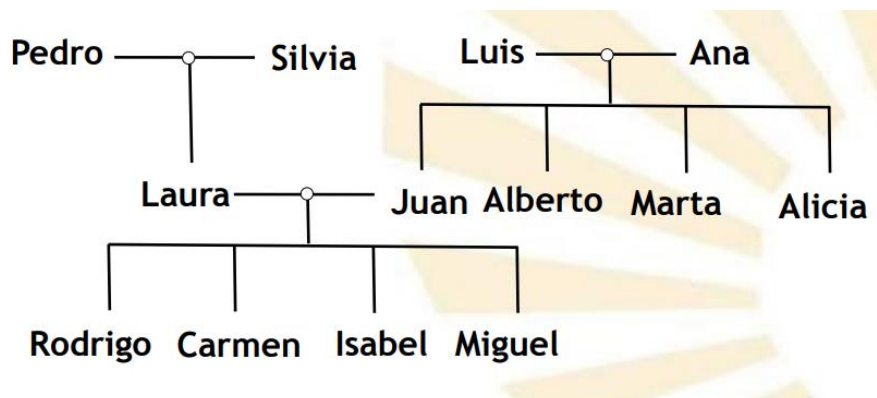
Parte III. Programación lógica con el lenguaje de programación Prolog.

1. Define los siguientes conceptos:

- Hechos.
- Reglas
- Consultas.

3. Resuelve ejercicios de programación en Prolog, tal como el siguiente:

Ejemplos de uso de reglas, variables, operadores usando un árbol genealógico. Con base en el siguiente diagrama:



Implementar los hechos y las reglas de modo que las siguientes consultas sean posibles:

% Definición de algunos hechos:

`padres(laura,pedro,silvia).`

...

% Ejemplos de consultas:

`hermana(alicia, alberto).`

`true.`

`hermana(alicia, laura).`

`false.`

`hermana(juan, alicia).`

`false.`

Hacer reglas para: ancestros, abuelo, abuela, primo, prima, tío, tía, etc. Se deben utilizar variables lógicas, operadores lógicos (AND, OR, NOT), con base en los hechos previamente definidos.

Parte IV. Programación Orientada a Objetos (POO)

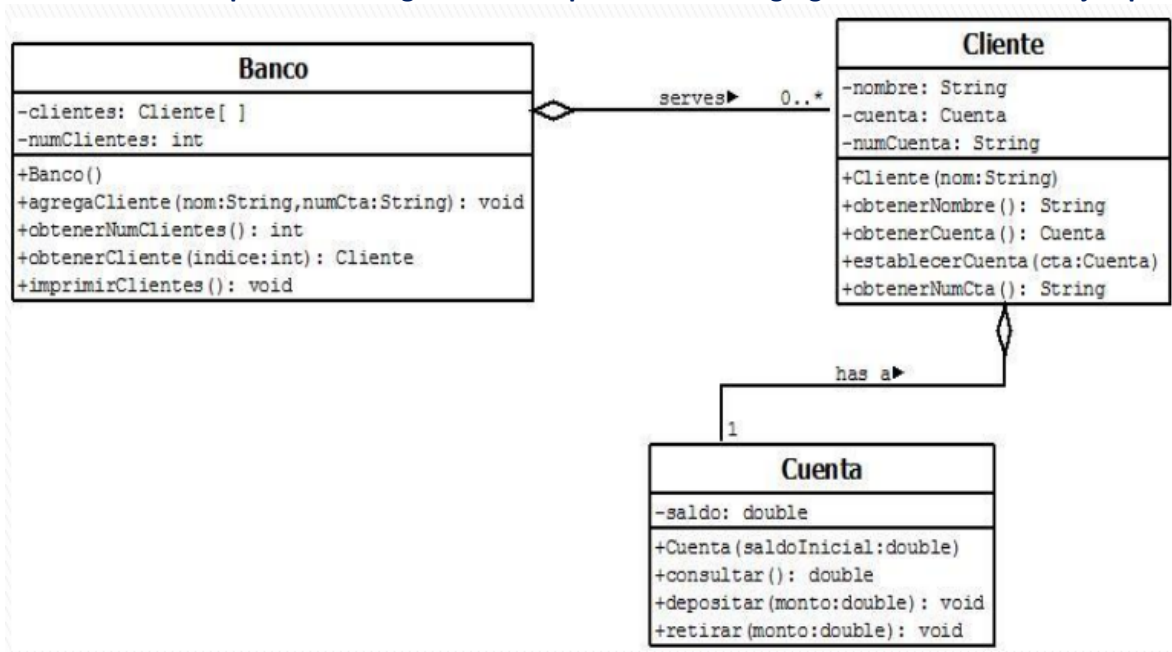
1. Define los siguientes conceptos:

- Principio de abstracción
- Principio de encapsulamiento
- Principio de modularidad
- Principio de jerarquía
- Principio de tipificación
- Clase y objeto
- Constructor (en el lenguaje de programación Java: constructor por omisión y constructor definido por el programador).
- Atributos y métodos.
- Sobrecarga de métodos.
- Variables de instancia y variables de clase

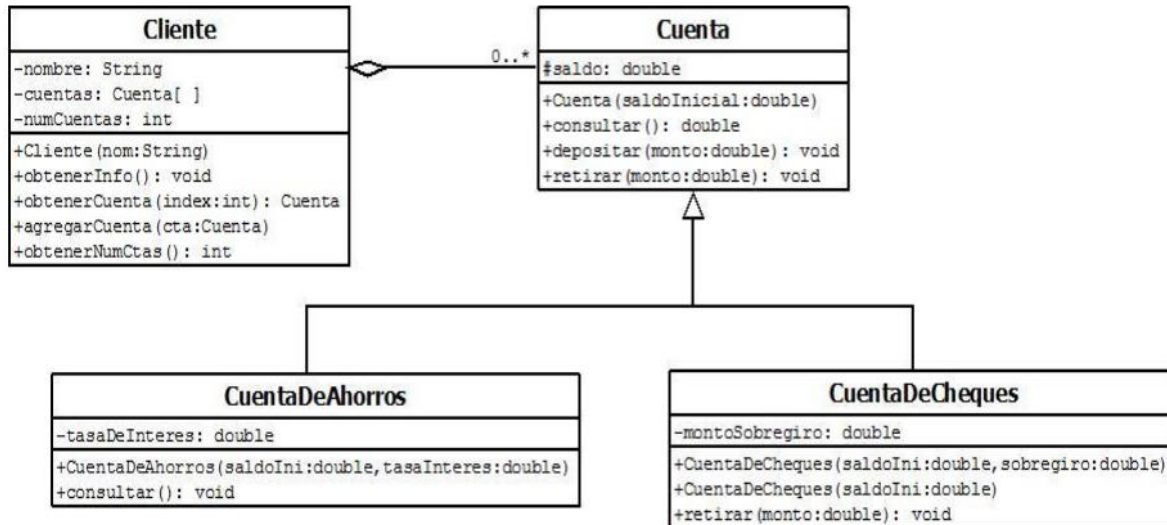
2. POO utilizando el lenguaje de programación Java:

- Manejo de variables referencia
- Uso de tipos de acceso (*private*, *public*, paquete (por omisión o *default*), *protected*)
- Clase Object.
 - Realizar la implementación de los métodos: `toString`, `equals`, dada una clase.

- **Uso de arreglos estáticos de objetos.**
 - Implementar el código para: encontrar un elemento (objeto) dentro de un arreglo, ordenar un arreglo, comparar dos arreglos
- **Uso de la clase ArrayList.**
 - Implementar el código para: encontrar un elemento (objeto) dentro del ArrayList, ordenar un ArrayList, comparar dos ArrayList
- **Manejo de static**
 - Implementar el código para: atributos, métodos y bloques estáticos.
- **Agregación de clases.**
 - Implementar diagramas UML que modelan la agregación de clases. Por ejemplo:



- **Herencia de clases**
 - Implementar la sobrescritura de métodos (*overridden*).
 - Implementar la especialización de clases agregando métodos nuevos en las clases derivadas.
 - Revisar el concepto de herencia directa: cuáles son los componentes que se heredan de la clase base a la clase derivada.
 - Implementar el tipo de acceso *protected*
 - Manejo de constructores en una jerarquía de clases
 - Implementar aplicaciones que usan agregación de clases y herencia de clases a la vez. Por ejemplo:



- Excepciones
- Definir el concepto de Excepciones verificadas y Excepciones no verificadas
- Implementar el manejo de excepciones (bloque *try-catch*)
- Ejercicios de manejo de excepciones
- Ilustrar el uso de las siguientes excepciones en código Java:

Excepciones verificadas	Excepciones No Verificadas
ClassNotFoundException	ArithmeticException
CloneNotSupportedException	ArrayIndexOutOfBoundsException
FileNotFoundException	ArrayStoreException
IllegalAccessException	ClassCastException
InstantiationException	IllegalArgumentException
InterruptedException	IndexOutOfBoundsException
IOException	NegativeArraySizeException
NoSuchMethodException	NullPointerException
	NumberFormatException
	StringIndexOutOfBoundsException

- Polimorfismo y forzamientos (para este apartado se puede tomar como base de implementación el diagrama UML anterior).
 - Implementar código para métodos polimórficos
 - Forzamiento

- Implementar código para el forzamiento hacia arriba (*upcasting*)
 - Implementar código para el forzamiento hacia abajo (*downcasting*)
- Clase abstracta
 - Implementar métodos abstractos de clases abstractas
 - Implementar jerarquías de clase con una clase base abstracta.
- Interfaces
 - Interfaces como contratos
 - Interfaces y jerarquías de clases
 - Interfaces y polimorfismo

