|  |  |  |  |
| --- | --- | --- | --- |
| **Nombres** |  | **Calificación** |  |
| **Fecha** |  |

1. **Definir lo siguiente:**

|  |  |
| --- | --- |
| **Clase** | Definición de las propiedades y comportamiento de un tipo de objeto en concreto. |
| **Herencia** | Es el proceso mediante el cual un objeto de una clase adquiere propiedades definidas en otra clase que lo preceda en una jerarquía de clasificaciones |
| **Polimorfismo** | Permite que un método tenga múltiples implementaciones, que se seleccionan en base al tipo objeto indicado al solicitar la ejecución del método. |
| **Objeto** | Es una entidad que posee propiedades o atributos y de comportamiento o funcionalidad. Es una instancia de una clase. |
| **Encapsulamiento** | Empaquetamiento de los elementos de una misma entidad. |
| **Ocultamiento** | Objetos aislados desde el exterior, permite la protección de los atributos del objeto. |

1. **Indicar cuantos constructores puede tener una clase y para qué sirven.**

Los que se considere necesario.

Inicializan los objetos en su clase.

1. **Indicar cuantos destructores puede tener una clase y para qué sirven.**

Uno.

Desempeña la tarea de limpieza de los objetos de su clase.

1. **Indicar cuál de las siguientes definiciones de servicios son correctos:**
2. insertarRegistroCliente()
3. crearNuevoCliente()
4. crearNuevoCliente (rut, nombre, apellidos, email, fono, direccion)
5. crearNuevoCliente (objetoCliente)
6. crearNuevoClientePsoft (schemaOracle, registroTablaCliente )
7. crearNuevoCliente (objetoCliente)
8. consultaUF()
9. **Indicar cuáles son las características que diferencian a un objeto de otro**
10. Métodos
11. Variables
12. Color
13. Herencia
14. Atributos
15. **Consiste en implementar múltiples formas de un mismo método**
16. Abstracción
17. Herencia
18. Multicast
19. Objeto
20. Polimorfismo
21. **¿Cuál es la forma correcta de declarar un vector?**
22. int vector [] = new int[10];
23. int vector [][] = new int [5][5];
24. int vector = new int[10];
25. int vector [] = new int[10]
26. int vector = new int[10]
27. **¿Cuál es la forma de comunicación entre objetos?**
28. Mensajes
29. Castings
30. Celular
31. Mail
32. Métodos
33. **¿En Java cuales de las siguientes afirmaciones son verdaderas?**
34. Es posible definir arrays bidimensionales de la forma a[i][j], donde cada fila a[i] puede tener distinta longitud.
35. No es posible declarar arrays con memoria estática
36. “Hola es un array”
37. “Hola es un objeto”
38. Un array es un objeto
39. **¿Cuál es el eje básico del modelo de POO?**
40. Herencia
41. Objeto
42. Clase
43. Polimorfismo
44. Todas las anteriores
45. **Responder correcto “C” / false “F”**

|  |  |
| --- | --- |
| Clase es la definición de las propiedades y comportamiento de un tipo de objeto en concreto. | V |
| Clase es el proceso mediante el cual un objeto de una clase adquiere propiedades definidas en otra clase que lo preceda en una jerarquía de clasificaciones. | F |
| Permite que una herencia tenga múltiples implementaciones, que se seleccionan en base al tipo objeto indicado al solicitar la ejecución del método. | F |
| Castings es la forma de comunicación entre objetos. | F |
| El principio de ocultamiento permite aislar los objetos desde el exterior y permite la protección de los atributos del objeto. | V |
| Una entidad que posee propiedades o atributos y de comportamiento o funcionalidad se la conoce como objetos. | V |
| El polimorfismo permite que un método tenga múltiples implementaciones, que se seleccionan en base al tipo objeto indicado al solicitar la ejecución del método. | V |
| El principio de ocultamiento permite que un método tenga múltiples implementaciones, que se seleccionan en base al tipo objeto indicado al solicitar la ejecución del método. | F |
| Una clase puede tener varios constructores | V |
| Una clase puede tener varios destructores | F |

1. **De este programa, cuál es la salida en pantalla:**

class A {

static int n = 0;

String f (A x) { return "AA" + (x.n++); }

String f (B x) { return "AB" + (x.n--); }

}

class B extends A {

String f (B x) { return "BB" + (x.n--); }

public static void main (String args[]) {

A a = new A ();

A b = new B ();

System.out.println (a.f (b));

System.out.println (b.f (b));

System.out.println (a.f ((B) b));

System.out.println (b.f ((B) b));

}

}

**Salida:**

*AA0*

*AA1*

*AB2*

*BB1*

1. **Definir una clase Conjunto que contenga:**

- Un array de valores de cualquier tipo.

- Un método intersección que tome como argumento otro conjunto, y devuelva un nuevo conjunto con la intersección de los dos.

class Conjunto {

Object elementos[];

Conjunto (Vector elems) {

elementos = new Object [elems.size ()];

for (int i = 0; i < elementos.length; i++)

elementos [i] = elems.elementAt (i);

}

Conjunto interseccion (Conjunto conj) {

Vector inter = new Vector ();

for (int i = 0; i < elementos.length; i++)

for (int j = 0; j < conj.elementos.length; j++)

if (elementos[i].equals (conj.elementos[j])) {

inter.addElement (elementos[i]);

break;

}

return new Conjunto (inter);

}

}

1. **Del siguiente programa.**

class A {

static int x[ ] = {1, 2};

int y[ ];

void f (int z[ ]) { z[1]+=2; }

void g ( ) {

A a = new A ( );

a.x[0]++;

}

}

class B {

public static void main (String args[ ]) {

A b = new A ( );

b.y = b.x;

b.f (b.y);

b.g ( );

System.out.println (b.x[0] + " " + b.x[1]); **(1)**

System.out.println (b.y[0] + " " + b.y[1]); **(2)**

}

}

**Indicar la salida de la línea 1:**

1. 1 2
2. 1 4
3. 2 2
4. 2 4
5. 2 1

**Indicar la salida de la línea 2:**

1. 1 2
2. 1 4
3. 2 2
4. 2 4
5. 2 1