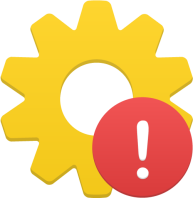
**CFP 8**

**Módulo 2:**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Unidad 5:**

**POLIMORFISMO**

****

## EJERCICIOS OBLIGATORIOS

### Polimorfismo, Diferencia entre polimorfismo y sobrecarga, Clases abstractas, Métodos abstractos, Interfaz, Herencia múltiple

1. Los métodos sobrescritos deben tener el mismo tipo de retorno y argumentos.

* Verdadero
* Falso

1. ¿Cuál es la salida del siguiente código?

**public** **class** Animal {

**public** **void** hacerSonido() {

System.***out***.println("Grrr...");

}

}

**public** **class** Gato **extends** Animal{

**public** **void** hacerSonido() {

System.***out***.println("Miau");

}

}

**public** **class** Ej02 {

**public** **static** **void** main(String[] args) {

Gato g = **new** Gato();

g.hacerSonido();

}

}

1. ¿A qué concepto se refiere la siguiente definición: “Una llamada a un método causará una implementación diferente a ser ejecutada, dependiendo del tipo de objeto que invoque al método.”?

* Polimorfismo
* Abstracción
* Encapsulamiento
* Variable

1. **Polimorfismo** es:

* Una implementación con diferentes métodos.
* Un método con diferentes implementaciones.
* Cada implementación, con un método diferente.

1. Cuando los métodos tienen diferente nombre, pero iguales parámetros, es conocido como **sobrecarga de métodos**.

* Verdadero
* Falso

1. Un **método abstracto** es un método que es declarado sin una implementación (sin llaves, y seguidas por un punto y coma). Ejemplo: **abstract** **void** caminar();
2. Una clase que contiene un **método abstracto** es una **clase abstracta**.

* Verdadero
* Falso

1. Completa los espacios en blanco para definir la **clase** Animal como **abstracta**, el **método** hacerSonido también como **abstracto** y para que la clase Gato herede de la clase Animal y defina el método hacerSonido() para dicha subclase.

**public** **abstract** **class** Animal {

**private** **int** patas;

**public** **abstract** **void** hacerSonido();

}

**public** **class** Gato **extends** Animal{

**public** **void** hacerSonido() {

System.***out***.println("Miau");

}

}

**Opciones posibles:** abstract, implements, extends, toString(), hacerSonido(),Abstract, private

1. Completa los espacios en blanco para crear una **clase abstracta** con un **método abstracto** y que la clase Perro herede de ella.

**public** **abstract** **class** Animal {

**protected int** edad;

**public** **abstract** **int** getEdad();

}

**public** **class** Perro **extends** Animal {

**public** **int** getEdad() {

**return** edad;

}

}

1. Una clase puede heredar de sólo **una** superclase, pero puede implementar **múltiples interfaces**.

* Verdadero
* Falso

1. En Java, ¿cuántas superclases puede tener tu subclase heredera?

* Ninguna
* Sólo una
* Solamente dos
* Múltiples

1. Arrastra y suelta las opciones a continuación para implementar una **interfaz**.

**public** **interface** IAnimal {

**public** **void** comer();

}

**public** **class** Pajaro **implements** IAnimal {

**public** **void** comer() {

System.***out***.println("Pájaro come");

}

}

**Opciones posibles:** IAnimal, abstract, Pajaro, return, class, void