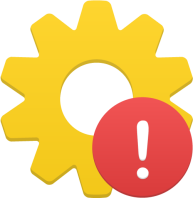
**CFP 8**

**Módulo 2:**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Unidad 3:**

**MÁS ACERCA DE CLASES**

****

## EJERCICIOS OBLIGATORIOS

### Relaciones entre clases, paquetes, equals, Array List, Enumerados, Iteradores

1. ¿Cuántos paquetes pueden estar contenidos en un programa Java?

* Tantos como se necesiten
* Ninguno
* Uno

1. El siguiente ejemplo muestra como utilizar la clase Vehiculo del paquete ejemplos:

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

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

Vehiculo v1 = **new** Vehiculo();

v1.tocarBocina();

}

}

* Verdadero
* Falso

1. ¿Cuál es la salida de este código?

**public** **class** A {

**private** **int** x;

**public** A() {

}

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

A a = **new** A();

a.x = 5;

A b = **new** A();

b.x = 5;

System.***out***.println(a == b);

}

}

* false
* true

1. Arrastra y suelta las opciones a continuación para validar si los dos objetos de tipo A son iguales en relación a su contenido.

**public** **class** A {

**private** **int** x;

**public** A() {

}

Public void setX(int x){

this.x = x;

}

Public int getX(){

return this.x;

}

**public** **boolean** equals(Object o) {

**A a = (A)o;**

**if(this.getX() == a.getX()){**

**return true;**

**}**

**Return false;**

}

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

A a = **new** A();

a.x = 9;

A b = **new** A();

b.x = 5;

System.***out***.println(a.equals(b));

}

}

**Opciones posibles:** new, b, boolean, equals, x, return, A

1. Los enumerados son utilizados para declarar variables que representan…

* Miembros de un conjunto prefijado
* Clases
* Interfaces
* Enteros

1. Arrastra y suelta las opciones a continuación para crear un enumerado llamado “Color”, con los valores ROJO, AZUL, VERDE.

**public** **enum** Color {

***ROJO***, ***AZUL***, ***VERDE***;

}

**Opciones posibles:** VERDE, class, ROJO, AZUL, enum, static

1. ¿Cuál es la salida de este código?

ArrayList<String> lista = **new** ArrayList<String>();

lista.add("A");

lista.add("B");

lista.add("C");

System.***out***.println(lista.get(1));

Salida : “B”

1. El método add() elimina objetos del ArrayList. En contraste, el método remove() añade nuevos objetos al ArrayList.

* Verdadero
* Falso

1. ¿Cuál es la salida de este código?

ArrayList<Integer> lista = **new** ArrayList<Integer>();

lista.add(10);

lista.add(20);

lista.add(30);

Iterator<Integer> it = lista.iterator();

it.next();

System.***out***.println(it.next());

Salida = 20

1. Completa los espacios en blanco para imprimir todos los elementos de la lista.

ArrayList<String> lista = **new** ArrayList<String>();

lista.add("A");

lista.add("B");

lista.add("C");

Iterator<String> it = lista.iterator();

**while** (it.hasNext()) {

System.***out***.println(it.next());

}

**Opciones posibles:** String, hasNext, Iterator, next, iterator