**Material de Trabajo Autónomo - Clases e Instancias**

La capacidad de poder definir una clase y lograr que esta ejecute funcionalidad es el primer paso para poder programar. Basados en lo visto en la primera unidad se pide desarrollar:

**Caso 1:**

Desarrollar una clase llamada Alumno que permita al alumno calcular su promedio de notas finales. Los alumnos están definidos por su código de alumno, nombre, apellido, notas: práctica 01, práctica 02, práctica 03, parcial y final. Esto da como resultado la nota final del alumno, que es lo que se quiere calcular, según esta fórmula:

PF = Promedio prácticas \* 0.2 + Parcial \* 0.3 + Final \* 0.5

Luego de creada la clase se pide instanciar 3 alumnos y que muestre a cada uno sus notas.

TIPS:

No olvidar que los atributos son privados y por lo tanto se les tiene que hacer accesibles con attr\_accessor

En caso quieras trabajar las notas como un arreglo puedes usar esta documentación <http://ruby-doc.org//core-2.2.0/Array.html> (no es indispensable para el ejercicio)

**Caso 2:**

Un casino desea desarrollar un canódromo virtual. Para ello quieren definir a cada uno de los perros corredores. Estos llevan como identificación un nombre, un número de pista (del 1 al 7) y una velocidad base (entre 1 y 10). Se necesita que el perro pueda mostrar sus datos de la siguiente manera:  
  
Nombre: <<el nombre del perro>> Número: <<el número del perro>>

En el caso de la velocidad se requiere que se cree un método llamado velocidad\_carrera que permita calcular la velocidad para una carrera particular. Cada vez que el perro corre puede generar una velocidad diferente. Para ello la fórmula es la siguiente:

Velocidad de Carrera = velocidad base + número aleatorio entre 1 y 10

Para generar el número aleatorio se pide investigar cómo se puede lograr que se genere y con ello desarrollar la fórmula.

TIPS:

Se puede utilizar esta documentación para investigar la generación de valores aleatorios mediante random <http://ruby-doc.org//core-2.2.0/Random.html>

Tener en cuenta que los valores de la velocidad son números enteros.

**Caso 3:**

Una empresa distribuidora de golosinas al por menor requiere un sistema que le permita realizar los cálculos de los totales a pagar de las facturas con mayor precisión debido a problemas de redondeo que ha tenido con SUNAT. Para ello se pide que se cree una clase llamada Boleta que tenga como atributo el monto. Adicionalmente se necesita que tengan dos métodos, el primero de ellos debe poder calcular el IGV mientras que el segundo deberá mostrar el total a pagar (igv calculado + monto original)

Sin embargo el total a pagar se debe redondear a dos decimales.

TIPS:

Se puede investigar el uso de round para redondear aquí: <http://ruby-doc.org/core-2.1.2/Float.html>

**Caso 4:**

Una empresa de arquitectura está implementando un sistema de presupuestos automáticos. Para ello requiere que la presentación muestra un mensaje de esta forma:

Carlos Perez, el presupuesto calculado es: 2983.12

Este mensaje debe ser mostrado en una sola cadena.

Para ello se pide crear una clase cuyos atributos sean el nombre de la persona y el monto a pagar.

TIPS :

Se puede investigar la conversión de valores flotantes a cadenas (string) aquí:

<http://ruby-doc.org/core-2.1.2/Float.html>

**Caso 5:**

Una empresa de dulces desea determinar el descuento a aplicar en una venta según el tipo de cliente. Para ello tiene tres tipos de clientes: cliente oro, plata y normal.

En el caso del cliente tipo oro el descuento es del 20% por pedidos de más de 1000 soles, en otro caso no hay descuento. En el caso de los clientes plata el descuento es de 10% por pedidos de más de 1200 soles, en otros caso no hay descuento. En el caso de los clientes normales no hay descuento en ningún caso.

Se pide crear el programa que permita calcular el descuento según el tipo de cliente.

**Caso 6:**

Un cliente de una empresa de telefonía ha realizado una compra de un celular de dicha empresa:

Los datos del cliente son: Nombre, dirección, DNI.

Del celular: número de celular, código de ciudad y saldo.

Se desea realizar una recarga de celular y actualizar su saldo.

Realice un programa orientado a objetos identificando las clases que intervendrían en la solución del problema.

a. Registre los datos de un cliente, y los datos del celular que posee.

b. Recargar el saldo de un celular de un cliente, muestre todos sus datos, el saldo inicial y el saldo final luego de la recarga.

**Caso 7:**

Un cambista de Ocoña necesita saber cuántos billetes de 100, 50 y 20 dólares respectivamente necesita para realizar un cambio de dinero de un cliente que le ha pedido le cambie una cantidad en soles.

Realice un programa orientado a objetos tal que le permita atender a dicho cliente.

Considere registrar los datos del cliente: Nombre y dni además del monto que desea que le cambien.

Considerar el tipo de cambio = 3.12 soles/dólar.

**Caso 8:**

Una veterinaria que baña perros requiere una programa que le permita desarrollar lo siguiente:  
  
Los perros deben ser registradas con un nombre, una raza y un tamaño. El servicio de baño tiene un costo base de 10 soles. Adicionalmente se tiene un parámetro según el tamaño del perro. En caso de ser pequeño tenemos un factor de 1 por lo que el precio será de 10 soles. En caso de un tamaño mediano el perro tiene un factor de 1.2 y en caso sea grande el factor del costo será de 1.5.

Por otro lado se requiere saber si el perro está vacunado. El costo base es de 20 soles. En caso sea perro pequeño el factor de costo será de 1.2, en caso se mediano de 1.8 y en caso sea grande de 2.0.

Se requiere que se registre al perro. En caso de no estar vacunado se deberá agregar el costo de vacuna pues es obligatorio y el del baño. En caso el perro esté vacunado solamente será el costo del baño. Si un perro pasa dos veces por el proceso de baño, la segunda vez ya no tendrá el costo de la vacuna pues ya se le ha vacunado en la primera visita.

**Caso 9:**

Desarrolle sus clases en Ruby que permita realizar el siguiente programa:

auto = Auto.new(“Volvo”, 19,000)

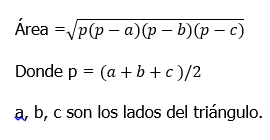
neto = auto.calcular\_neto(2)

puts “Neto: #{neto}”

donde el método calcular\_neto(2), debe calcular el costo de dos autos Volvo con precio 19,000 y además le suma IGV que es una constante 0.18.

5- Un Matemático desea calcular el área de un triángulo dado sus lados, realice un programa que permita ingresar los lados del triángulo e imprima su área.

El área de un triángulo se calcula así:



**Caso 10:**

Diseñe una clase que identifique en su centro de labores y pueda resolver un problema o una necesidad.

Impleméntelo en Ruby.

Considere los datos de entrada y salida.