Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI–3661 – Laboratorio de Lenguajes de Programación.
Enero–Marzo 2020

Tarea III: Ruby

(10 pts)

Implementación:

1. (3 pts) – Propiedades y herencia.

   - Considere una clase `Vector2`, representando elementos de $\mathbb{R}^2$, con dos campos, `x, y` correspondiente a las coordenadas del vector.

     a) (0.25 pts) – Defina la clase en cuestión, con ambos campos.

     b) (0.25 pts) – Implemente *setters* y *getters* para ambos campos. Implemente un método `initialize` (constructor) que reciba dos números e inicialice las coordenadas `x, y` del vector.

     c) (0.25 pts) – Implemente un método `magnitud` que calcule la magnitud de un vector. Use como norma la distancia euclídea entre las dos coordenadas del vector (la definición usual de norma).

     d) (0.5 pts) – Implemente métodos para sumar, restar, igualdad y multiplicar (producto punto/escalar) dos `Vector2`. Debe implementarlos como operadores, usando $+$, $-$, $==$ y $*$ respectivamente. Implemente además un método para $-$ como operador unario, de manera que devuelva una copia del mismo vector con los signos de sus coordenadas invertidos.

     e) (0.25 pts) – Implemente el método `to_s` de manera que cualquier vector pueda ser representado en una cadena de caracteres como (x, y).

   - Los números complejos $\mathbb{C}$ tienen una interpretación geométrica en $\mathbb{R}^2$, tomando la parte real como la coordenada x y la parte imaginaria como la coordenada y. Considere una subclase `Complejo` de `Vector2`..

     a) (0.25 pts) – Defina la subclase en cuestión. Debe redefinir el método `initialize` para que el valor de la parte imaginaria (y) sea 0 por defecto.

     b) (0.75 pts) – Redefina el método de multiplicación ($*$) para reflejar las reglas de multiplicación de complejos. Implemente además, usando $\sim$ como operador unario, un método que devuelva el conjugado de un número complejo (es decir, debe retornar una copia del mismo número, con el signo de la parte imaginaria invertido). Luego, implemente, usando el operador /, un método para la división de dos números complejos. En caso de que se divida por cero, debe atajar la excepción e imprimir "No se puede dividir entre 0".

     c) (0.5 pts) – Implemente el método `to_s` de manera que ahora la representación en cadena de caracteres sea la usual para números complejos. Es decir:
        - Si ambas partes son positivas: $x + yi$
        - Si la parte real es negativa: $-x + yi$
        - Si la parte imaginaria es negativa: $x - yi$

- Si ambas partes son negativas: $-x - yi$
- Si la parte imaginaria es cero: x
- Si la parte real es cero: yi

2. (2.5 pts) – Defina una clase `Moneda` con subclases `Dolar`, `Yen`, `Euro`, `Bolivar` y `Bitcoin`.

   a) (0.5 pts) – Defina métodos `dolares`, `yens`, `euros`, `bolivares` y `bitcoins` sobre la clase `Float` que convierta el flotante en dólares, yens, euros, bolívares y bitcoins, respectivamente.

   b) (1 pt) – Defina un método `en` sobre la clase `Moneda` (y sus subclases, por ende) que reciba un átomo entre `:dolares`, `:yens`, `:euros`, `:bolivares` y `:bitcoins` y convierta la moneda en aquella representada por el átomo propuesto.
   Por ejemplo: `15.dolares.en(:euros)` debe evaluar en `12.72` euros.

   c) (1 pt) – Defina un método `comparar` sobre la clase `Moneda`, que reciba otra `Moneda` y las compare.
   - Debe devolver `:menor` si la primera moneda es menor que el argumento.
   - Debe devolver `:igual` si la primera moneda es igual que el argumento.
   - Debe devolver `:mayor` si la primera moneda es mayor que el argumento.

   Por ejemplo: `100000.bolivares.comparar(2.dolares)` debe evaluar en `:menor`.

   Nota: Use doble despacho para averiguar los tipos del argumento pasado. No pregunte por el tipo explícitamente.

3. (1.5 pts) – Bloques e iteradores.

   Dadas dos colecciones (de tipos posiblemente diferentes), se desea calcular el producto cartesiano de los elementos generados para cada una de ellas.

   Por ejemplo: El producto cartesiano de `[:a, :b, :c]` y `[4, 5]` debe generar:

   - `[:a, 4]`
   - `[:a, 5]`
   - `[:b, 4]`
   - `[:b, 5]`
   - `[:c, 4]`
   - `[:c, 5]`

   Nota 1: No importa el orden en que se devuelvan los elementos, sino que todos los elementos aparezcan.

   Nota 2: El elemento `[:a, 4]` *está en el resultado del ejemplo anterior, pero* `[4. :a]` *no. El orden interno de las tuplas es importante.*

Investigación:

Considere una lenguaje de programación puramente orientado a objetos, donde una clase B hereda de otra clase A (esto es, B es subclase de A).

a) (1.5 pts) – Considere una clase `Lista`, parametrizable en el tipo de sus elementos. ¿Qué relación de herencia, de haberla, tienen `Lista<`A`>` y `Lista<`B`>`?

- (0.5 pts) – ¿Qué decisión toma el lenguaje *Java*? Explique dicha decisión.
- (0.5 pts) – ¿Qué decisión toma el lenguaje *Scala*? Explique dicha decisión.
- (0.5 pts) – Suponiendo que existe una clase `Bottom`, la cual es subclase de todas las demás clases: ¿Cuál es el tipo inferido de la lista vacía? Justifique su respuesta.

b) (1.5 pts) – Considere ahora que el lenguaje de programación en el que se está trabajando es funcional. Como es puramente orientado a objetos, las funciones también deben ser objetos. Suponga otra clase cualquiera C.

- (0.75 pts) – ¿Qué relación de herencia, de haberla, tienen las funciones con firmas $A \to C$ y $B \to C$? Justifique su respuesta.
- (0.75 pts) – ¿Qué relación de herencia, de haberla, tienen las funciones con firmas $C \to A$ y $C \to B$? Justifique su respuesta.

.

Detalles de la Entrega:

La entrega de la tarea consistirá de un único archivo `t3_<carné1>&<carné2>.pdf`, donde `<carné1>` y `<carné2>` son los números de carné de los integrantes de su equipo. Tal archivo debe ser un documento PDF con su implementación para las funciones pedidas y respuestas para las preguntas planteadas. Por ejemplo, si el equipo está conformado por `00-00000` y `11-11111`, entonces su entrega debe llamarse: `t3_00-00000&11-11111.pdf`.

La tarea deberá ser entregada a *ambos* profesores encargados del curso (Carlos Infante y Alexander Romero), *únicamente* a sus direcciones de correo electrónico institucionales: (13-10681@usb.ve y 13-11274@usb.ve) a más tardar el Viernes 6 de Marzo, a las 11:59pm. VET.
