Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI–3661 – Laboratorio de Lenguajes de Programación.
Septiembre–Diciembre 2024

Tarea 2: Programación Orientada a Objetos y Ruby

(20 pts)

Implementación:

1. (4 puntos) – Propiedades y herencia.

   - Considere una clase `Circulo`, con un único campo `radio`.

     a) Defina la clase en cuestión, con el campo propuesto.

     b) Implemente *setters* y *getters* para el campo `radio` de `Circulo`.

     c) Implemente un método `initialize` (constructor) para `Circulo` que reciba un número e inicialice el radio del círculo con dicho número. En caso de que el número propuesto sea negativo, se debe arrojar un error con el mensaje: `'Radio invalido'`.

     d) Implemente un método `area` para `Circulo` que retorne el área del círculo.

   - Considere una subclase `Cilindro` de `Circulo`, que agrega un único campo `altura`.

     a) Defina la subclase en cuestión, con el campo adicional propuesto.

     b) Implemente *setters* y *getters* para el campo `altura` de `Circulo`.

     c) Implemente un método `initialize` (constructor) para `Cilindro` que reciba dos números e inicialice el radio y altura del cilindro con dicho número. En caso de que el radio propuesto sea negativo, se debe arrojar un error con el mensaje: `'Radio invalido'`. En caso de que la altura propuesta sea negativo, se debe arrojar un error con el mensaje: `'Altura invalida'`.

     d) Implemente un método `volumen` para `Cilindro` que retorne el volumen del cilindro.

2. (4 puntos) – Defina una clase `Moneda` con subclases `Dolar`, `Yen`, `Euro`, `Bolivar` y `Bitcoin`.

   a) 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) 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 `14.16` euros.

   c) 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: `50.bolivares.comparar(2.dolares)` debe evaluar en `:menor`.

   Nota: Investigue de qué trata el doble despacho y úselo para averiguar los tipos del argumento pasado. No pregunte por el tipo explícitamente.

3. (4 puntos) – 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:

1. (4 puntos) – 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) Considere una clase `Lista`, parametrizable en el tipo de sus elementos. ¿Qué relación de herencia, de haberla, tienen `Lista<`A`>` y `Lista<`B`>`?

      - ¿Qué decisión toma el lenguaje *Java*? Explique dicha decisión.

      - ¿Qué decisión toma el lenguaje *Scala*? Explique dicha decisión.

      - 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) 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.

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

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

   .

2. (4 puntos) – En Ruby existen dos tipos de jerarquía: una basada en *herencia* y otra basada en *instanciación*.

   - Si `A` es *subclase* de `B`, entonces `A.superclass` será `B`. Esto establece la jerarquía de *herencia*.
   - Si `A` es una *instancia* de `B`, entonces `A.class` será `B`. Esto establece la jerarquía de *instanciación*.

   Por ejemplo:

   - La clase para el valor `42`, que se puede consultar como `42.class` es `Integer` (ya que `42` es un entero).
   - La clase para el valor `Integer`, que se puede consultar como `Integer.class` es `Class` (ya que `Integer` es una clase).
   - La superclase para el valor `Integer`, que se puede consultar como `Integer.superclass` es `Numeric` (ya que todo entero es un valor numérico).

   Tomando esto en cuenta:

   a) ¿Cuáles son las raíces para las jerarquías de herencia e instanciación?

   b) Si `A` y `B` son las raíces encontradas para las jerarquías, respectivamente:
      - ¿Cuál es el resultado de aplicar `A.class`?
      - ¿Cuál es el resultado de aplicar `A.superclass`?
      - ¿Cuál es el resultado de aplicar `B.class`?
      - ¿Cuál es el resultado de aplicar `B.superclass`?

   c) Supongamos que existe un valor `A` tal que `A.class = A`. Si es cierto, nos interesan todos los demás valores que no cumplen con esa propiedad.

      Definimos un valor compuesto por todos aquellos valores que no son instancias de si mismos:

      $$R = \{A \mid A.class \neq A\}$$

      Notemos, por ejemplo, que $42 \in R$, ya que `42.class = Integer`.

      Si consideramos `R` como un valor más, ¿cuál debería ser el resultado de aplicar `R.class`? ¿Será cierto que $R \in R$?

   d) Explique la relación que tiene el resultado anterior con la paradoja de Russell.

Detalles de la Entrega

La entrega de la tarea consistirá de un único archivo PDF con su implementación para las funciones pedidas y respuestas para las preguntas planteadas.

La tarea deberá ser entregada al prof. Ricardo Monascal *únicamente* a su dirección de correo electrónico: (rmonascal@gmail.com) a más tardar el Lunes 25 de Noviembre , a las 11:59pm. VET.