**Universidad de La Laguna. Escuela Técnica Superior de Ingeniería Informática**
**Tercero del Grado de Informática**
**LENGUAJES Y PARADIGMAS DE PROGRAMACION. SEGUNDA PARTE**
5 páginas

Nombre: _____ Alu: _____

1. ¿Cuál es la visibilidad del método `initialize`?

2. El valor retornado por `initialize` es usado para la construcción del objeto. ¿Verdadero o falso?

3. Considere el siguiente código *Ruby*:

```
1   class AClass
2     @x = 4
3     @y = 9
4
5     def initialize(x,y)
6       @x, @y = x, y
7     end
8   end
```

   *a*) ¿Qué tipo de variable son las de las líneas 2 y 3? ¿Cuál es su visibilidad? Explique su respuesta.

   *b*) ¿Qué tipo de variable son las de la línea 6? ¿Cuál es su visibilidad? Explique su respuesta.

4. Señale el error de concepto en la codificación de la siguente clase *Ruby*:

```
class MyClass
  def initialize(b)
    @b = b
  end
  def to_s
    puts @b
  end
end
```

5. Considere el siguiente código *Ruby*:

```
1   class AClass
2     attr_accessor :n
3     ...
4     def initialize(n)
5       @n = n
6     end
7     ...
8     def square
9       n = n * n
10    end
11    ...
12  end
```

   En los accesos a `n` a la derecha de la igualdad en la línea 9

   *a*) ¿a qué tipo de variable se está accediendo?

*b*) ¿y en el acceso a la izquierda de la igualdad?

*c*) ¿`attr_accessor` es un método de instancia o de clase?

*d*) ¿En qué clase está definido `attr_accessor`?

6. ¿Qué ventajas e inconvenientes se tienen si en la escritura de un método `metodo(x,y)` se usa `respond_to?` para comprobar que los argumentos `x` e `y` pueden ser utilziados dentro del cuerpo de `metodo`?

7. En *Ruby* ¿el conocimiento de la clase `obj.class` del objeto `obj` caracteriza la conducta del objeto?

8. ¿A qué clase pertenece el objeto que crea la llamada `Fraction = Struct.new(:num, :denom)`?

9. ¿Cómo se puede impedir que se invoque a los métodos `num=` y `denom=` de la clase `Fraction = Struct.new(:num, :denom)`?

10. ¿En qué forma se define en *Ruby* un método de clase?

11. ¿Es posible definir una constante de la clase `MyClass` antes de la definición del método `initialize`?

12. ¿Es posible definir constantes de una clase desde fuera de la misma?

13. ¿A qué clase de variable `n` permite acceder la declaración de la línea 3?

```
1       class Tutu
2         class << self
3             attr_accessor :n
4         end
5       end
```

14. ¿Qué tipo de herencia proporciona *Ruby*?

15. ¿Qué puede ocurrir si en una subclase `A` se escribe un método con nombre `intimo` igual al de un método privado `intimo` de su superclase `B`? ¿Qué ocurre si el método `toto` de la superclase llama a `intimo` con destino un objeto de la clase `A`?

16. ¿Qué ocurre cuando se llama a `super` sin argumentos?

17. ¿Cómo se puede llamar a `super` sin argumentos?

18. Suponga que la clase `B` hereda de `A` un método `tutu` que usa la constante `C` definida en `A`. Si en la clase `B` se define `C`, ¿Qué definición de `C` usará `tutu`, la de `A` o la de `B`?

19. ¿Cuál es la visibilidad por defecto de un método?

20. ¿Cuál es la visibilidad por defecto de un método que ha sido definido fuera de cualquier clase (por ejemplo en un script)?

21. Los métodos privados no pueden ser llamados desde otra clase que no sea aquella en la que se declararon, ¿cierto o falso?

22. Dentro de una clase y fuera de un método `self`, ¿a qué objeto hace referencia `self`?

23. Un método de instancia de la clase `Class` es un método ___ _____ del objeto de la clase `Class`

24. El módulo `Math` permite el acceso de dos formas:

```
[~/rubytesting/TheRubyProgrammingLanguage/Chapter7ClassesAndModules]$ irb
ruby-1.9.2-head :001 > Math.sin(Math::PI/2)
 => 1.0
ruby-1.9.2-head :002 > include Math
 => Object
ruby-1.9.2-head :003 > sin(PI/2)
 => 1.0
```

¿Cómo se crea un módulo que funcione de esta manera?

25. ¿Qué diferencias hay entre los siguientes predicados?

    *a*) `==`
    *b*) `eql?`
    *c*) `equal?`
    *d*) `===`
    *e*) `=~`

26. ¿Cuál es el resultado?

```
> (1..10) === 5
=> ???
> /\d+/ === "123"
=> ???
> String === "s"
=> ???
> :s === "s"
=> ???
```

27. ¿Cómo se puede permitir que los objetos de la clase `Fraction = Struct.new(:num, :denom)` sean comparables?

28. ¿Qué predicado es usado por *Ruby* para comprobar la igualdad entre claves de un `hash`?

29. ¿Cómo se puede conseguir qué el producto de un número por un objeto de una clase que se está definiendo funcione? Por ejemplo: `4 * obj`

30. ¿En que clase se define `protected`? ¿Es un método de instancia o de clase?

31. ¿Disponen los elementos de la clase `Module` de un método `new`? ¿Dispone la clase `Module` de un método `new`?

32. ¿Cómo se puede hacer que la única forma de construir objetos de la clase `MyClass` se haga mediante nuestro propio método factoría/constructor `my_maker` (desde una clase externa a `MyClass`)?

33. ¿Qué es una clase abstracta? ¿Cómo se define una clase concreta?

34. ¿Qué comentario al comienzo del fichero permite usar caracteres UTF-8 dentro del programa?

35. ¿En qué directorio hay que ubicar las pruebas unitarias?

36. ¿Qué ficheros se han de requerir para implementar las pruebas unitarias?

37. ¿Cómo se denomina la clase *Ruby* de la cual hay que heredar para implementar las pruebas?

38. Describa el comportamiento de la afirmación `assert_raise(exception_type,..){<code block}` y proponga un ejemplo de uso.

39. ¿Cómo se llaman los métodos que permiten la factorización de código que debe ejecutarse al principio y al final de cada prueba unitaria?

40. ¿Qué opción permite ejecutar los test unitarios cuyos nombres concuerdan con un patrón?

41. Escriba una tarea de `Rake` para lanzar las pruebas unitarias.

42. ¿Cómo se denomina el paradigma de desarrollo en el que se basa la herramienta `RSpec`?

43. Describa el conjunto de pasos a seguir para desarrollar una aplicación con `RSpec`.

44. ¿En qué directorio hay que implementar las especificaciones de los requerimientos a una clase?

45. Rellene las partes que faltan de esta especificación `Rspec`:

```
1  class RSpecGreeter
2    def greet
3      "Hello RSpec!"
4    end
5  end
6
7  describe "RSpec Greeter" do
8    it "should say 'Hello RSpec!' when it receives the greet() message" do
9      greeter = RSpecGreeter.new
10     greeting = greeter.greet
```

```
11      greeting._____ == "_____________________"
12    end
13
14    it "says 'Hello [something]'" do
15      greeter = RSpecGreeter.new
16      greeting = greeter.greet
17      greeting._____ _____ /^Hello [a-zA-Z]+/
18    end
19  end
```

46. ¿Como puedo factorizar las líneas

    ```
    greeter = RSpecGreeter.new
    greeting = greeter.greet
    ```

    en el programa del ejercicio 45 anterior? ¿que cambios debo introducir en el programa de especificación?

47. ¿Qué opción permite ejecutar `RSpec` de manera que muestre por consola la descripción de cada una de las especificaciones realizadas?

48. Escriba una tarea de `Rake` para lanzar las pruebas de `RSpec`.

49. ¿En qué consiste la integración continua (*Continuous Integration*)?

50. Describa el conjunto de pasos a seguir para desarrollar una aplicación con `Travis`.

51. ¿En qué directorio hay que implementar los requerimientos de `Travis`?

52. ¿Qué se ha de especificar en el fichero `.travis.yml`?

53. Describa el contenido de un fichero `Gemfile` utilizado por `Travis`.

54. Escriba un fichero `Rakefile` que permita utilizar `Travis`.

55. ¿Cómo se pone en funcionamiento `Travis`?

56. Rellene el siguiente *Ruby koan* (el método `instance_variables` retorna un `Array` con las variables de instancia del objeto):

    ```
    class Dog2
      def set_name(a_name)
        @name = a_name
      end
    end

    def test_instance_variables
      fido = Dog2.new
      assert_equal __________, fido.instance_variables
      fido.set_name("Fido")
      assert_equal __________, fido.instance_variables
    end
    ```