1. Un equipo de desarrollo ha decidido comenzar a trabajar en forma un poco mas cuantitativa al momento de hacer estimaciones. El tiempo de desarrollo estimado de un proyecto sigue una distribución normal, pero han decidido aproximarlo por una triangular.

La estimación del próximo proyecto es que de ninguna manera tomará menos de 10 ni más de 110 días con el valor mas probable en 36 días.

a) El contratista quiere que el proyecto se termine en a lo más 90 días. ¿Cual es la probabilidad de que el proyecto acabe en ese lapso?

b) El jefe del equipo le indica al contratista que puede ser posible entregar el proyecto en la mitad del tiempo de los 90 días pero la probabilidad es baja ¿Cual es esa probabilidad?

Solución

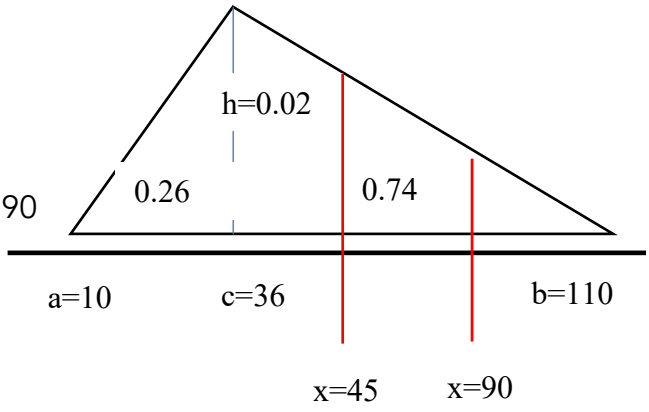
Se trata de una distribución triangular en que a = 10, b= 110 y c = 36

Lo primero es normalizar la distribución de modo que el área total bajo la curva sea 1.

Supongamos que la altura del triángulo es h, entonces

(c-a)*h/2 + (b-c)*h/2 = 1

26h/2 + 74h/2 = 1 => h = 0.02



a) Corresponde al área a la izquierda de x=90

Eso es 1 - el pequeño triángulo a la derecha de x=90

La altura hx se puede calcular por

h/74 = hx/20 => hx = 20*0.02/74 = 0.0054

1 - 20*0.0054/2 = 0.946

La probabilidad es de un 94.5%

b) Corresponde al área a la izquierda de x=45

Eso es 1 - el triángulo a la derecha de x=45

La altura hx se puede calcular por

h/74 = hx/65 => hx = 65*0.02/74 = 0.0175

1 - 65*0.0175/2 = 0.431

La probabilidad es de un 43.1%

Notas de Corrección

Este problema es bastante simple y se requiere que los números estén correctos. Si por ejemplo el alumno olvida normalizar (cálculo de h) tendrá todo malo en adelante.

No se acepta el uso de una distribución normal en lugar de una triangular.

2. Se quiere construir una aplicación que permita mantener información del DCC desplegada en una gran pantalla a su entrada en el cuarto piso. La información que se presenta corresponde a últimas publicaciones de los profesores, conferencias en que han participado, premios recibidos, noticias de prensa, actividades sociales, etc.

La información la ingresan primero los profesores y alumnos de postgrado y luego son revisadas y ajustadas/corregidas por la encargada de difusión. En algunos casos la publicación puede quedar pendiente para consultas con algún profesor.

La encargada de difusión debe poder agregar y eliminar cosas de lo que se muestra y controlar el orden y el tiempo en que aparecen.

a) (1.5 pts) Dibuje un diagrama de casos de uso identificando actores y casos de uso

b) (1.5 pts) Describa en forma detallada 3 de los principales casos de uso

Solución

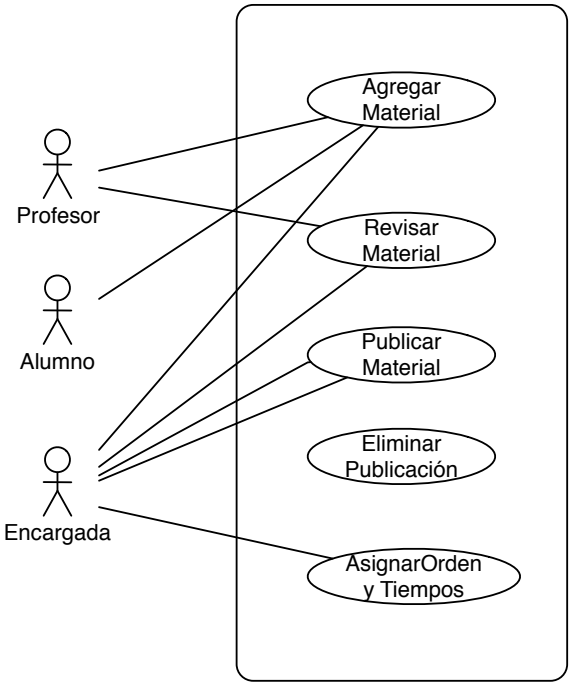
a) Actores

- Profesor
- Alumno
- Encargada

Casos de Uso

1. Someter un nuevo material a publicación
2. Revisar un material para ser publicado
3. Publicar un material revisado
4. Eliminar un material publicado
5. Asignar el orden y los tiempos de exposición de cada material publicado

Diagrama de Casos de Uso



Los principales casos de uso son los primeros 3 (pueden ser cualquiera)

b)

UC1: Caso de Uso: Agregar Material

Actores Primarios: Profesor, Alumno, Encargada

Precondiciones: El actor dispone del material preparado para ser subido al sistema

Postcondiciones: El nuevo material queda subido en un área para ser revisado

Flujo Principal

1. El profesor/alumno/encargada indica al sistema que quiere subir un nuevo material para ser publicado

2. El sistema entrega opción de subir material desde el computador local

3. El profesor/alumno/encargada elige el documento

4. El sistema pide confirmación

5. El profesor/alumno/encargada confirma

6. El material queda guardado en área de revisión

Flujos alternativos

2a. El sistema entrega opción para subir material desde una URL

2a1. El actor ingresa una URL válida

2a2. El flujo regresa al paso 4

5a El actor se arrepiente y no confirma

5a1. El flujo regresa al paso 3

6a. El material no es encontrado

6a1. El sistema indica que no se encontró el documento en el pathname o url

6a2. El flujo regresa al 3

6b El material es de un formato no aceptado

6b1. El sistema indica que solo se aceptan contenidos en formatos pdf, html, jpg, png, ...

6b2. El flujo regresa a 3

UC2: Revisar Material para publicación

Actores Primarios: Encargada

Precondiciones: El material se encuentra disponible en área de revisión

Postcondiciones: El nuevo material queda aprobado para ser publicado

Flujo Principal

1. La encargada indica que quiere revisar los materiales disponibles para revisión
2. El sistema le presenta la lista de materiales no revisados ordenados por fecha
3. La encargada selecciona un material de la lista
4. El sistema despliega el material y da la opción de editar
5. La encargada indica que ha terminado de editar
6. El documento se guarda en estado de revisado

Flujos Alternativos

2a. No hay materiales para revisión

2a1. El sistema indica que no hay materiales disponibles para revisión y el caso de uso termina

6a El documento no puede ser guardado (falla de formato)

6a1. El sistema da una opción de corregir e intentar nuevamente

6a2. Si se presenta el mismo problema el flujo vuelve a 3

UC3: Publicar Material

Actores Primarios: Encargada

Precondiciones: El material se encuentra en estado aprobado

Postcondiciones: El nuevo material queda publicado en el sistema

Flujo Principal

1. La encargada indica que quiere publicar un material
2. El sistema le presenta la lista de materiales en estado revisado
3. La encargada selecciona un material de la lista
4. El sistema despliega el material y pide confirmación
5. La encargada confirma publicación
6. El material queda publicado

Flujos Alternativos

2a. No hay materiales en estado revisado

2a1. El sistema indica que no hay materiales disponibles para publicar y el caso de uso termina

Notas de Corrección

Dado que en el curso se da mucho mas énfasis a los relatos de usuario es posible que algunos alumnos escriban relatos en lugar de casos de uso. En ese caso la nota no puede ser otra que el mínimo.

Lo más importante en cada caso de uso es la descripción del flujo principal y al menos un flujo alternativo. El uso de la numeración es importante. Cada flujo alternativo debe comenzar con la condición que desencadena este flujo.

3. Considere una función de validación de un módulo de software para un supermercado que debe aceptar el nombre de un ítem y una lista de los diferentes tamaños (máximo 5) en que se ofrece expresados en onzas (números enteros de 1 a 48). El nombre debe ser alfabético de 2 a 15 caracteres. La lista de tamaños debe ser ingresada en orden ascendente. El nombre del ítem se ingresa primero seguido de una coma y luego la lista de tamaños separados por comas. Los espacios en blanco son ignorados. Por ejemplo 'Chips_Lays', 1, 2, 4, 10. La función devuelve un boolean que indica si el input es correcto (True) o incorrecto (False)

Diseñe cuidadosamente un set de pruebas mínimo que permita testear este software usando técnicas de caja negra.

Solución

Identificamos las clases de equivalencia

1. Nombre es alfabético
2. Nombre es no alfabético
3. Nombre es de menos de dos caracteres
4. Nombre es de 2 a 15 caracteres
5. Nombre es de más de 15 caracteres
6. tamaño menor que 1
7. tamaño entre 1 y 48
8. tamaño mayor que 48
9. tamaño que es número entero
10. tamaño que es un número no entero
11. tamaño que no es número
12. Tamaños en orden ascendente
13. Tamaños en desorden o descendente
14. No hay tamaños
15. Uno a 5 tamaños
16. Más de 5 tamaños
17. Nombre del item primero
18. Nombre del item no es el primero
19. Una coma separa cada elemento
20. Elementos sin coma separadora
21. Espacios en blanco sobrantes
22. Sin espacios en blanco sobrantes

Diseñamos ahora casos de prueba representantes de ellos. Cada caso de prueba debe especificar el valor esperado de retorno. Al menos debe haber un representante de cada clase (algunos pueden pertenecer a más de una clase)

Caso	Input	Output
1	xy,1	T
2	4AbcDefghijklmn,1,2,3 ,4,48	F
3	x,4	T
5	Abcdefghijklmnop, 8, 12	F
6	Xyz, 0	F
7	XY ,47	T
8	Xy, 52, 60	F
9	Xy,1, 2, 4, 8	T
10	Xy, 2.5	F
11	XY,2,4,o,7	F
12	Xyz,4,6,8	T
13	XY , 2,3,4,1,6	F
14	AB	F
15	Abc,1,3,5,8,10	T
16	Abc,1,3,5,8,10,12	F
17	Abc, 3, 6	T
18	1, xyz	F
19	Abc,1,3,5	T
20	Abc,135	F
21	Xy, 1,2 4	T
22	xyz,2,4,8	T

Notas de Corrección

Lo más importante es identificar las clases de equivalencia. Puede que se les pasen algunas pero deben estar la mayoría de ellas.

Al diseñar los casos es importante que en cada uno de ellos aparezca un input y el output esperado.

4. Se quiere simular el funcionamiento de varios posibles negocios de venta de comida: una pizzería, una hamburguesería y una ensaladería. Para simplificar supongamos que en cada caso solo se fabrican dos productos:

- Pizzería: pizza de pepperoni y pizza vegetariana
- Hamburguesería: regular y not_meat
- Ensaladería: cesar y mediterranea

Queremos sacar partido del patrón Abstract Factory y para ello se pide

a) Definir tres fábricas abstractas: una para las pizzas una para las hamburguesas y una para las ensaladas. Además de escribir el código de las fábricas escriba el código de los productos (lo más simple posible)

b) Definir una clase Negocio con un método llamado simular que recibe una fábrica y el número de productos de cada tipo y procede a hacer la simulación como muestra el ejemplo. Su código debe funcionar exactamente de la misma manera.

Solución

a) Las fábricas son las clases Pizzeria, Hamburguesería y Ensaladería
Los productos concretos son las clases Pepperoni, Vegetarian, Regular, Not_meat, Cesar y Mediterranean.

```
class Pizzeria
    def new_p1(number)
        Pepperoni.new(number)
    end
    def new_p2(number)
        Vegetarian.new(number)
    end
end
class Hamburgueseria
    def new_p1(number)
        Regular.new(number)
    end
    def new_p2(number)
        Not_meat.new(number)
    end
```

```
end
class Ensaladeria
    def new_p1(number)
        Cesar.new(number)
    end
    def new_p2(number)
        Mediterranean.new(number)
    end
end
```

Los productos concretos son las clases Pepperoni, Vegetarian, Regular, Not_meat, Cesar y Mediterranean.

```
class Pepperoni
    def initialize (number)
        @name = 'pizza peperoni ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
class Vegetarian
    def initialize (number)
        @name = 'pizza vegetariana ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
class Regular
    def initialize (number)
        @name = 'hamburguesa regular ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
class Not_meat
    def initialize (number)
        @name = 'hamburguesa not_meat ' + number
    end
```

```
    def reveal
        return @name + ' saliendo'
    end
end
class Cesar
    def initialize (number)
        @name = 'ensalada cesar ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
class Mediterranean
    def initialize (number)
        @name = 'ensalada mediterranea ' + number
    end
    def reveal
        return @name + ' saliendo'
    end
end
```

b) La clase Negocio se inicializa con una fábrica y una cantidad de cada uno de los dos productos. Esta clase incluye el método simular.

```
class Negocio
    def initialize(meal_factory, number_product_1, number_product_2)
        @the_factory = meal_factory
        @p1s = []
        number_product_1.times do |i|
            p1 = @the_factory.new_p1("#{i+1}")
            @p1s << p1
        end
        @p2s = []
        number_product_2.times do |i|
            p2 = @the_factory.new_p2("#{i+1}")
            @p2s << p2
        end
      end
    def simular
        @p1s.each {|p1| puts(p1.reveal)}
```

```
            @p2s.each {|p2| puts(p2.reveal)}
        end
    end

    (Negocio.new(Hamburgueseria.new,3,1)).simular
    (Negocio.new(Pizzeria.new,2,3)).simular
    (Negocio.new(Ensaladeria.new,4,1)).simular
```

Notas de Corrección

Lo más importante es el uso correcto del patrón fábrica abstracta. Cada una de las 3 fábricas deben tener dos métodos para producir cada uno de sus dos productos.

Es importante que el funcionamiento debe ser el indicado, es decir:

(Negocio.new(Hamburgueseria.new,3,1)).simular

```
hamburguesa regular 1 saliendo
hamburguesa regular 2 saliendo
hamburguesa regular 3 saliendo
hamburguesa not_meat 1 saliendo
```

(Negocio.new(Pizzeria.new,2,3)).simular

```
pizza peperoni 1 saliendo
pizza peperoni 2 saliendo
pizza vegetariana 1 saliendo
pizza vegetariana 2 saliendo
pizza vegetariana 3 saliendo
```

(Negocio.new(Ensaladeria.new,4,1)).simular

```
ensalada cesar 1 saliendo
ensalada cesar 2 saliendo
ensalada cesar 3 saliendo
ensalada cesar 4 saliendo
ensalada mediterranea 1 saliendo
```

5. Estás haciendo tu práctica en la empresa TodosOnline, una nueva red social. El siguiente código escrito por un compañero de trabajo se encarga de hacer una llamada a una API de la compañía y procesar los datos obtenidos. La llamada a la API es de tipo GET a la url https://internalapidata.todosOnline.com/dev/contacts. Exige una autenticación por medio de JWT, y da resultados paginados mediante el parámetro de query "page". (Por ejemplo: https://internalapidata/.todosOnline.com/dev/contacts?page=1 trae todos los contactos de la página 1).

La API devuelve datos tipo JSON de contactos con la estructura que se ve en el

archivo api-response.json descargar. Cada objeto identifica a un contacto con un id, su nombre, apellido, edad y de quién es amigo. Para esto último muestra un arreglo con los ids de contacto de sus amigos.

El código procesa esta respuesta y retorna un arreglo de hashes con la información de

los contactos como se ve en el archivo final-data.rb descargar. En donde cada contacto se identifica por nombre completo, edad, si es mayor de edad y una lista con los nombres completos de sus amigos.

El código hace dos llamadas para las primeras dos páginas de información. Actualmente está funcionando, pero le diste un vistazo al código y es un dolor de cabeza.

a) (10 pts) Identifica 3 code smells que haya en el código. Nombra cuáles son, en qué parte del código se encuentra (Puedes mencionarlo o poner el código) y explicar cómo se puede solucionar ese code smell.

b) (10 pts) Tu compañero escucha eso de code smells y te dice "Si tienes tan buen olfato, haz que huela bien" y te pasa la responsabilidad. Haz un refactor del código solucionando los code smells y mejorando lo que encuentres, eso incluye mejorar nombres, agregar comentarios y mejorar performance si es posible.

El formato de la respuesta es texto en un PDF. Puedes usar un editor de código si lo deseas, pero debes copiar y pegar el código en un PDF. No se revisarán archivos .rb, .json u otra extensión que no sea pdf. El código se encuentra en El código se encuentra en el archivo code.rb

Solución y pauta

1. a)

- Existe código repetido en ambas llamadas a la API, toda la lógica del método process_data está dos veces. Para solucionar eso, el código que se repite se puede colocar en un método y llamar a ese método las veces que sean necesarias.
- ProcessData es una clase muy grande. La clase tiene más de una responsabilidad, hacer la llamada a la API y procesar los datos. Esto se puede solucionar descomponiendo la clase en dos clases, cada una con una única responsabilidad.
- Existe una lazy class que corresponde a la clase CheckTrue. Esta clase no está haciendo nada útil. No es necesario una clase para checar el valor de un boolean. Para solucionarlo se puede eliminar la clase y mover su responsabilidad a una evaluación de valor simple.
- Existe el code smell Message Chains en la invocación de los métodos construct_url, add_stage y add_endpoint. Los métodos intermedios no están agregando nueva funcionalidad, lo que provoca una seguidilla de invocaciones innecesarias. Para solucionarlo se puede eliminar los métodos intermedios y tener un sólo método que construya la url completa.

Otro code smell que se puede identificar es long_method en el método process_data. Debido al código repetido este método tiende a ser muy largo, y si es necesario hacer un cambio se debe hacer en múltiples partes. La solución es la misma que el código repetido.

Pauta:

- 1 punto base, solo si contestó la pregunta.
- 1 punto por cada code smell correctamente reconocido en el lugar correcto. (3 en total)
- 2 puntos por cada code smell con la explicación correcta de cómo solucionarlo (6 en total)

b)

Además de los code smells identificados, vemos que los nombres de las variables no son explicativos, no existen comentarios y muchas veces se ocupan variables innecesarias. Por otro lado, vemos que por cada amigo de cada contacto se recorre

nuevamente la lista de contactos, es decir, tenemos 3 loops, uno dentro de otro que afecta al performance. El tercer loop se usa para encontrar los nombres de los contactos. Esta información se podría almacenar en un diccionario, el que se crea iterando una sola vez los contactos, y no por cada amigo de contacto. Para esto, construimos el diccionario fuera de los 2 loops principales.

Pauta

- 1 punto por cambios de nombres a las variables poco explicativas (r2j, r2k, r, r2, etc…) y agregar comentarios útiles.
- 2 puntos por eliminar código repetido.
- 2 puntos por separar la clase en dos subclases, o bien, separar el método process_data según responsabilidades (llamar a la API y procesar los datos)
- 2 puntos por deshacerse de la clase CheckTrue
- 2 puntos por deshacerse de la cadena de llamadas para construir la url
- 1 punto por usar el diccionario u otra estructura de datos para eliminar la complejidad de los loops.