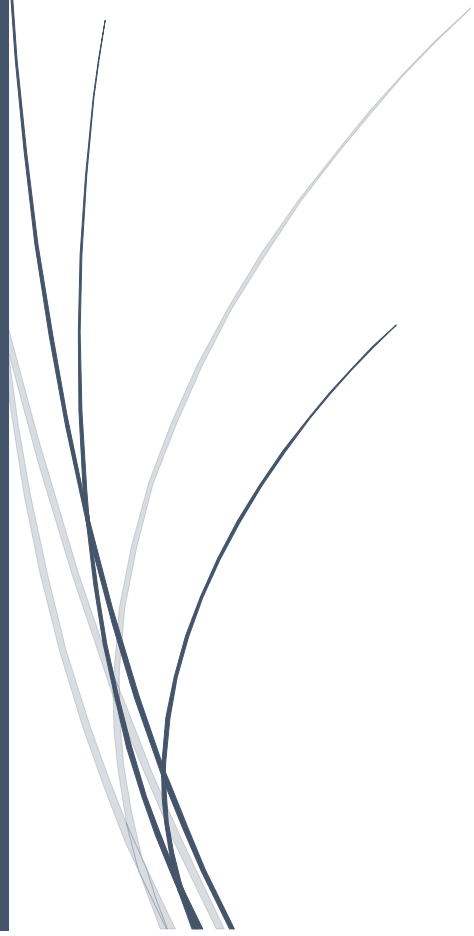
22-12-2023

Ejercicio teórico 1

Ingeniería del Software II



MIGUEL GÓMEZ GÓMEZ-LOBO
JOSE FRANCISCO GÓMEZ – CALCERRADA ÁLVAREZ

Introducción

Está wiki está destinada para el ejercicio número 1 del trabajo teórico. Este ejercicio será desarrollado por `Jose Francisco Gómez-Calcerrada Álvarez` y `Miguel Gómez Gómez-Lobo`

Enunciado del Ejercicio

Se trata de escribir y probar una clase Persona que inicialice objetos con estados válidos. Los datos que se deben tomar de la Persona deben ser nombre, apellidos, fecha de nacimiento, nacionalidad, titulación, certificación de inglés, número de teléfono, y correo electrónico. Además, teniendo en cuenta la semántica del problema nos interesa incluir métodos que nos ayuden a determinar si la persona es mayor de edad, si es Europea o no y si puede matricularse en un programa de doctorado porque tenga una titulación igual o superior a un máster. Valore en la medida de lo posible, la gestión y uso de excepciones, así como la posibilidad de crear un programa donde se use la clase persona. Tenga en cuenta, además de las consiguientes consideraciones semánticas del problema, los siguientes requisitos para el programa:

• La interfaz de usuario será en línea de comando. Se tienen que desarrollar clases con métodos específicos para leer de teclado y escribir en pantalla cadenas y números.

• Utilice excepciones en aquellos puntos del programa donde sea necesario controlar el flujo del programa.

• No acople las clases de dominio con interfaz de usuario (p.ej. no genere salidas por pantalla en las clases de la lógica de dominio)

Problemas a tratar

En el enunciado se nos enumeran varios puntos a elaborar, nosotros los hemos resumido en 5 apartados:

1. Escribir el código que se identifique en base al anterior enunciado.
2. Identificar las variables a probar del código e identificar sus valores de prueba.
3. Calcular el número máximo de casos de prueba y hacer dos conjuntos de prueba (each use y pairwaise).
4. Realizar un conjunto de prueba para alcanzar la cobertura de decisiones.
5. Realizar un conjunto de prueba para alcanzar la cobertura MC/DC.

1. Código

En este apartado de la wiki se puede observar la implementación del problema. Se ha dividido en 4 clases, la primera de ellas es el objeto Persona, donde se almacenarán los métodos principales. Después tendremos la excepción DatosNoValidosException y la interfaz IU_Usuario para controlar los errores, y finalmente tenemos la clase Main donde probaremos todas las clases anteriores.

Clase Persona

Esta clase cuenta con los métodos esMayorDeEdad para comprobar la mayoría de edad de la persona, esEuropeo para ver si su nacionalidad es europea, y puedeMatricularseDoctorado para comprobar si se puede matricular en un doctorado. En el caso del método de nacionalidad europea se ha simplificado la lógica de tal manera que en el atributo de nacionalidad tendría que aparecer la cadena europeo para no tener que mirar todos los paises. El código es el siguiente:

```
import java.time.LocalDate;
import java.time.Period;

public class Persona {
    private String nombre;
    private String apellidos;
    private LocalDate fechaNacimiento;
    private String nacionalidad;
    private String titulacion;
    private boolean certificacionIngles;
    private String numeroTelefono;
    private String correoElectronico;

    public Persona(String nombre, String apellidos, LocalDate fechaNacimiento,
                   String nacionalidad, String titulacion, boolean certificacionIngles,
                   String numeroTelefono, String correoElectronico) {
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.fechaNacimiento = fechaNacimiento;
        this.nacionalidad = nacionalidad;
        this.titulacion = titulacion;
        this.certificacionIngles = certificacionIngles;
        this.numeroTelefono = numeroTelefono;
        this.correoElectronico = correoElectronico;
    }

    public boolean esMayorDeEdad() {
        return Period.between(this.fechaNacimiento, LocalDate.now()).getYears() >= 18;
    }

    public boolean esEuropeo() {
        return this.nacionalidad.equalsIgnoreCase("europea");
    }
```

```
    public boolean puedeMatricularseDoctorado() {
        return this.titulacion.equalsIgnoreCase("master") ||
               this.titulacion.equalsIgnoreCase("doctorado");
    }

    //Getters y setters
        public String getNombre() {
                return nombre;
        }

        public void setNombre(String nombre) {
                this.nombre = nombre;
        }

        public String getApellidos() {
                return apellidos;
        }

        public void setApellidos(String apellidos) {
                this.apellidos = apellidos;
        }

        public LocalDate getFechaNacimiento() {
                return fechaNacimiento;
        }

        public void setFechaNacimiento(LocalDate fechaNacimiento) {
                this.fechaNacimiento = fechaNacimiento;
        }

        public String getNacionalidad() {
                return nacionalidad;
        }
```

```
        public void setNacionalidad(String nacionalidad) {
                this.nacionalidad = nacionalidad;
        }

        public String getTitulacion() {
                return titulacion;
        }

        public void setTitulacion(String titulacion) {
                this.titulacion = titulacion;
        }

        public boolean isCertificacionIngles() {
                return certificacionIngles;
        }

        public void setCertificacionIngles(boolean certificacionIngles) {
                this.certificacionIngles = certificacionIngles;
        }

        public String getNumeroTelefono() {
                return numeroTelefono;
        }

        public void setNumeroTelefono(String numeroTelefono) {
                this.numeroTelefono = numeroTelefono;
        }

        public String getCorreoElectronico() {
                return correoElectronico;
        }
```

```
        public void setCorreoElectronico(String correoElectronico) {
                this.correoElectronico = correoElectronico;
        }

}
```

Clase DatosNoValidosException

```
public class DatosNoValidosException extends Exception {
    public DatosNoValidosException(String mensaje) {
        super(mensaje);
    }
}
```

Clase IU_Usuario

```
import java.util.Scanner;

public class IU_Usuario {
    private Scanner scanner;

    public IU_Usuario() {
        this.scanner = new Scanner(System.in);
    }

    public String leerCadena(String mensaje) {
        System.out.println(mensaje);
        return scanner.nextLine();
    }

    public int leerEntero(String mensaje) throws DatosNoValidosException {
        System.out.println(mensaje);
        if (!scanner.hasNextInt()) {
            scanner.next(); // Limpiar el buffer
            throw new DatosNoValidosException("Se esperaba un número entero");
        }
        return scanner.nextInt();
    }

    public boolean leerBoolean(String mensaje) {
        System.out.print(mensaje);
        while (true) {
            String input = scanner.nextLine().trim().toLowerCase();
            if (input.equals("true") || input.equals("false")) {
                return Boolean.parseBoolean(input);
            } else {
                System.out.println("Entrada inválida. Por favor ingrese 'true' o 'false':");
            }
        }
    }
}
```

Clase Main

En esta clase probaríamos las anteriores:

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;

public class Main {
    private static IU_Usuario iu = new IU_Usuario();

    public static void main(String[] args) {
        try {
            Persona persona = crearPersonaDesdeConsola();
            mostrarInformacionPersona(persona);
        } catch (DatosNoValidosException | DateTimeParseException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    private static Persona crearPersonaDesdeConsola() throws DatosNoValidosException {
        String nombre = iu.leerCadena("Ingrese el nombre: ");
        String apellidos = iu.leerCadena("Ingrese los apellidos: ");
        LocalDate fechaNacimiento = leerFecha("Ingrese la fecha de nacimiento (formato YYYY-MM-DD): ");
        String nacionalidad = iu.leerCadena("Ingrese la nacionalidad: ");
        String titulacion = iu.leerCadena("Ingrese la titulación: ");
        boolean certificacionIngles = iu.leerBoolean("¿Tiene certificación de inglés? (true/false): ");
        String numeroTelefono = iu.leerCadena("Ingrese el número de teléfono: ");
        String correoElectronico = iu.leerCadena("Ingrese el correo electrónico: ");

        return new Persona(nombre, apellidos, fechaNacimiento, nacionalidad,
                           titulacion, certificacionIngles, numeroTelefono, correoElectronico);
    }

    private static LocalDate leerFecha(String mensaje) throws DatosNoValidosException {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        String fecha = iu.leerCadena(mensaje);
        try {
            return LocalDate.parse(fecha, formatter);
        } catch (DateTimeParseException e) {
            throw new DatosNoValidosException("Formato de fecha inválido.");
        }
    }

    private static void mostrarInformacionPersona(Persona persona) {
        System.out.println("\nInformación de la persona:");
        System.out.println("Es europeo: " + persona.esEuropeo());
        System.out.println("Es mayor de edad: " + persona.esMayorDeEdad());
        System.out.println("Puede matricularse en un programa de doctorado: " +
                           persona.puedeMatricularseDoctorado());
    }
}
```

2. Variables a probar

Variables de interés

Tras evaluar el código realizado, se ha observado que las variables más relevantes son los atributos nacionalidad y titulación pertenecientes al objeto Persona, y la variable edad, que viene de calcular edad de una persona comparando la fecha actual con la de hace 18 años, esto se consigue en la línea return Period.between(this.fechaNacimiento, LocalDate.now()).getYears() >= 18;. Aunque esta ultima variable no esté implicita en el código, se va a tomar como edad la parte izquierda de la comparación con el valor 18. Estas tres variables se han elegido puesto que son las que se utilizarán en los métodos esMayorDeEdad(), esEuropeo() y puedeMatricularseDoctorado(). La tabla con sus posibles valores es la siguiente:

Variable	Tipo	Rango
edad	Int	((-2^31)-1 , 2^31)
nacionalidad	String	∞
titulacion	String	∞

Clases de equivalencia

Primero de todo se obtienen las clases de equivalencia de cada una de las variables:

Variable	Clase de equivalencia
edad	(-∞, 0)[0, 18)[18, +∞)
nacionalidad	europea, no europea
titulacion	ninguna, educacion obligatoria, bachillerato, grado, master, doctorado

Valores límite de variante ligera

Para este caso se obtienen todos los valores de los límites de las clases de equivalencia, concretamente los valores que se encuentren en los extremos que sean abiertos:

Variable	Valores variante ligera
edad	0,18
nacionalidad	true, false
titulacion	ninguna, educacion obligatoria, bachillerato, grado, master, doctorado

Valores límite de variante pesada

Para este caso se obtendrán todos los valores superiores e inferiores de los valores obtenidos en la variante ligera:

Variable	Valores variante pesada
edad	-1,1,17,19
nacionalidad	
titulacion	

Conjetura de errores

Para este caso se obtendrán valores que estén dentro de las clases de equivalencia y pensemos que pueden ser subsceptibles de llevar a error.

Variable	Conjetura de errrores
edad	-(2^31)-1, 2^31, 5000
nacionalidad	null
titulacion	null

3. Conjuntos de prueba

Número máximo de casos de prueba

Para calcular el número máximo de casos de prueba se debe de multiplicar todos los valores posibles de cada una de las variables. Para la variable edad vamos a contar que podemos tener desde 0 hasta 250 años como máximo (siendo este un valor muy extremo), para la variable nacionalidad tenemos 2 valores posibles, y para titulación tenemos 6 posibles valores por lo tanto 251x2x6 nos da un total de 3012 casos de prueba posibles.

Conjunto de pruebas each use

Este conjunto de prueba consiste principalmente en utilizar al menos una vez cada valor en el caso de prueba. Como se observa en las tablas del punto 2, tenemos dos valores para nacionalidad, 5 valores para titulación y 9 valores para edad, por lo cual realizaremos 9 casos de prueba que es el mayor número de valores que nos encontramos entre las variables:

Variables	1	2	3	4	5	6	7	8	9
edad	0	18	-1	1	17	19	-(2^31)-1	2^31	5000
nacionalidad	europeo	no europeo	europeo	no europeo	europeo	no europeo	europeo	no europeo	europeo
titulacion	ninguna	educacion obligatoria	bachillerato	grado	master	doctorado	ninguna	grado	master

Conjunto de pruebas pairwaise

En este caso utilizaremos la misma tabla que hemos utilizado anteriormente para el conjunto de casos de prueba each use, pero en vez de escoger el valor mas alto, escogeremos los 2 valores mas altos y los multiplicaremos entre si para obtener el número total de casos de prueba:

9 * 5 = 45 casos

Nº	Edad	Nacionalidad	Titulación
1	0	Europeo	Ninguna
2	18	No Europeo	Educación Obligatoria
3	-1	Europeo	Bachillerato
4	1	No Europeo	Grado
5	17	Europeo	Máster
6	19	No Europeo	Doctorado
7	-(2^31)-1	Europeo	Ninguna
8	2^31	No Europeo	Grado

9	5000	Europeo	Máster
10	0	No Europeo	Educación Obligatoria
11	18	Europeo	Bachillerato
12	-1	No Europeo	Grado
13	1	Europeo	Máster
14	17	No Europeo	Doctorado
15	19	Europeo	Ninguna
16	-(2^31)-1	No Europeo	Grado
17	2^31	Europeo	Máster
18	5000	No Europeo	Educación Obligatoria
19	0	Europeo	Bachillerato
20	18	No Europeo	Grado
21	1	No Europeo	Ninguna
22	17	Europeo	Grado
23	19	No Europeo	Máster
24	-(2^31)-1	Europeo	Doctorado
25	2^31	No Europeo	Educación Obligatoria
26	5000	Europeo	Bachillerato
27	0	No Europeo	Grado
28	18	Europeo	Máster
29	-1	No Europeo	Doctorado
30	1	Europeo	Ninguna
31	17	No Europeo	Educación Obligatoria
32	19	Europeo	Bachillerato
33	-(2^31)-1	No Europeo	Grado
34	2^31	Europeo	Máster
35	5000	No Europeo	Doctorado
36	0	Europeo	Educación Obligatoria
37	18	No Europeo	Bachillerato
38	-1	Europeo	Grado
39	1	No Europeo	Máster
40	17	Europeo	Doctorado
41	19	No Europeo	Ninguna
42	-(2^31)-1	Europeo	Educación Obligatoria
43	2^31	No Europeo	Bachillerato
44	5000	Europeo	Grado
45	0	No Europeo	Máster

Comentario de los resultados

Apartado Nº Máx. Casos de prueba

En este apartado se calculaba el número máximo de casos de prueba que se puede realizar con nuestro código, al darnos un número tan elevado como es el 3012 nos es inviable realizar todos los casos distintos, por lo tanto se aplicarán las técnicas de each-use y pairwise para probar el código.

Apartado Each-Use

Se necesitan 9 casos de prueba, lo cual es un número manejable y razonable. La cobertura Each-Use ofrece un equilibrio entre la cantidad de pruebas y la cobertura alcanzada. Aunque puede dejar de lado algunos casos que podrían ser críticos, es un enfoque viable cuando el tiempo y los recursos son limitados. Sin embargo, hay que ser consciente del riesgo de que algunos escenarios menos comunes puedan no ser probados.

Apartado PairWise

Se obtuvieron 45 casos de prueba, lo que es un número moderado y factible para la implementación. La cobertura PairWise es más completa que Each-Use y ofrece un buen equilibrio entre cobertura y cantidad de casos. Es especialmente útil en situaciones donde las interacciones entre pares de variables son críticas. Dado que esta cobertura prueba todas las combinaciones posibles de pares de variables, es probable que se capturen la mayoría de los errores relacionados con interacciones inesperadas.

Conclusión

Entre las opciones presentadas, la cobertura PairWise parece ser la más equilibrada, ofreciendo una cobertura suficiente sin un número excesivo de casos de prueba. Es adecuada para garantizar que se prueben las interacciones críticas entre variables, sin la sobrecarga de MaxCdP. En escenarios donde el tiempo es un factor crítico y los recursos son limitados, Each-Use podría ser una opción adecuada, pero con la advertencia de que puede dejar algunos escenarios sin probar. La elección entre Each-Use y PairWise dependerá del nivel de riesgo que el equipo esté dispuesto a asumir y de los recursos disponibles para las pruebas.

4. Cobertura de decisiones

Para este apartado vamos a identificar todos los condicionales que nos encontramos en nuestro código:

Condicional esMayorDeEdad()

```
public boolean esMayorDeEdad() {
    return Period.between(this.fechaNacimiento, LocalDate.now()).getYears() >= 18;
}
```

El primero de los condicionales que nos encontramos es en el método que calcula si una persona es mayor de edad o no, concretamente está en la comparación entre la edad de la persona y el 18. Para obtener la cobertura de decisiones debemos de obtener la tabla de verdad de está condición, para posteriormente hacer un caso de prueba.

Condiciones		Edad	Nacionalidad	Titulación
T		20		
F		14		

Condicional esEuropeo()

```
public boolean esEuropeo() {
    return this.nacionalidad.equalsIgnoreCase("europea");
}
```

La segunda de las condiciones que encontramos en nuestro código pertenece al método que comprueba si una persona es europea o no, está condición está en el equals que compara la nacionalidad de la persona para ver si es europea. Al igual que en la condición anterior se obtendrán la tabla de verdad y los casos de prueba de la misma.

Condiciones		Edad	Nacionalidad	Titulación
T			europea	
F			no europea	

Condicional puedeMatricularseDoctorado()

```
public boolean puedeMatricularseDoctorado() {
    return this.titulacion.equalsIgnoreCase("master") ||
           this.titulacion.equalsIgnoreCase("doctorado");
}
```

Las últimas condiciones que encontramos están presentes en el método donde se comprueba si una persona puede matricularse en un doctorado. En esté caso como nos encontramos una comparación OR, tenemos por cada lado del equals una decisión. Como en los otros casos, se obtiene la tabla de verdad y los casos de prueba.

Condiciones		Edad	Nacionalidad	Titulación
T				master
F				grado

Condiciones		Edad	Nacionalidad	Titulación
T				doctorado
F				bachillerato

Casos de prueba finales

Tras obtener los casos de prueba para cada una de las condiciones es necesario unirlos para obtener un conjunto de casos de prueba general que satisfaga la cobertura de decisiones. El conjunto de casos de prueba es el siguiente:

Edad	Nacionalidad	Titulación izquierda	Titulación derecha
20	europeo	master	bachillerato
14	no europeo	grado	doctorado
20	europeo	master	doctorado
14	no europeo	grado	bachillerato

5. Cobertura MC DC

Para este apartado vamos a identificar de nuevo todos los condicionales que nos encontramos en nuestro código:

Condicional esMayorDeEdad()

```
public boolean esMayorDeEdad() {
    return Period.between(this.fechaNacimiento, LocalDate.now()).getYears() >= 18;
}
```

El primero de los condicionales que nos encontramos es en el método que calcula si una persona es mayor de edad o no, concretamente está en la comparación entre la edad de la persona y el 18. Para obtener la cobertura de decisiones debemos de obtener la tabla de verdad de está condición con el valor de la decisión para posteriormente hacer un caso de prueba.

Condiciones	Decisión		Edad	Nacionalidad	Titulación
T	T		20		
F	F		14		

Condicional esEuropeo()

```
public boolean esEuropeo() {
    return this.nacionalidad.equalsIgnoreCase("europea");
}
```

La segunda de las condiciones que encontramos en nuestro código pertenece al método que comprueba si una persona es europea o no, está condición está en el equals que compara la nacionalidad de la persona para ver si es europea. Al igual que en la condición anterior se obtendrán la tabla de verdad y los casos de prueba de la misma.

Condiciones	Decisión		Edad	Nacionalidad	Titulación
T	T			europea	
F	F			no europea	

Condicional puedeMatricularseDoctorado()

```
public boolean puedeMatricularseDoctorado() {
    return this.titulacion.equalsIgnoreCase("master") ||
           this.titulacion.equalsIgnoreCase("doctorado");
}
```

La última de las condiciones que encontramos está presente en el método donde se comprueba si una persona puede matricularse en un doctorado. En esté caso está presente una comparación OR, donde tenemos por un lado un equals que compara si en la titulación se encuentra algún master y por el otro tenemos un equals que compara si en las titulaciones encontramos un doctorado. Como en los otros casos, se obtiene la tabla de verdad y los casos de prueba.

Condicion izquierda	Condición derecha	Decisión		Edad	Nacionalidad	Titulación izquierda	Titulación derecha
T	T	T				master	doctorado
T	F	T				master	grado
F	T	T				bachillerato	doctorado
F	F	F				grado	grado

Casos de prueba finales

Tras obtener los casos de prueba para cada una de las condiciones es necesario unirlos para obtener un conjunto de casos de prueba general que satisfaga la cobertura de decisiones. El conjunto de casos de prueba es el siguiente:

Edad	Nacionalidad	Titulación izquierda	Titulación derecha
20	europeo	master	doctorado
14	no europeo	master	grado
20	europeo	bachillerato	doctorado
14	no europeo	grado	grado