Práctica UD 3.1

Diseño de pruebas

Desenvolvemento de Aplicacións Web

Contornos de Desenvolvemento

Sumario

Instrucciones

- Las capturas de las máquinas virtuales deben mostrar el nombre de la máquina.
- En el nombre de la máquina virtual debe contener la inicial y el apellido del alumno/a que entrega la práctica.
  - Por ejemplo, si creo una máquina virtual llamada "vsFTPd Server", debo nombrarla "jlopez vsFTPd Server".
- Las capturas deben de tener una calidad suficiente para que su contenido pueda ser legible.
- La entrega será en la tarea de la plataforma moodle mediante un fichero pdf practica_x_tu_nombre.pdf (x es número de practica (3.1) y tu_nombre es tu nombre) en el que se puedan ver en las diferentes secciones lo solicitado.

Objetivo de la Tarea

El objetivo de esta tarea es que demuestres tu capacidad para diseñar casos de prueba, aplicando técnicas de:

- Grafos de flujo y complejidad ciclomática.
- Clases de equivalencia.
- Análisis de valores límite (AVL).
- Conjetura de errores.

Todo ello se hará a partir del código fuente en java de un sistema de clasificación de usuarios que verifica según la edad si una persona está en edad de estudiar, de trabajar o de jubilarse.

Entorno de trabajo

Sistema Anfitrión: El de vuestro equipo.

Sistema Huésped (Virtualizado): Máquina virtual con un sistema operativo linux (Debian, Unbuntu…) y un IDE que permita compilar y ejecutar el código java.

Entrega

1. Crea un documento PDF (Nombrado como: Tarea3_1_TuNombre_TuApellido.pdf).

2. Responde a las cuestiones

3. Autoevalúate en la siguiente rúbrica

Rúbrica de autoevaluación

Tarea	OK / KO	Comentarios
Parte 1.1 (2 puntos) Grafo de Flujo y Complejidad Ciclomática		
Parte 1.2 (2 puntos) Clases de Equivalencia		
Parte 1.3 (2 puntos) Análisis de Valores límite		
Parte 1.4 (2 puntos) Conjetura de errores		
Parte 1.5 (2 puntos) Tabla de Casos de Prueba		
	NOTA FINAL:	

0. Prerrequisitos

De momento este es un ejercicio teórico que no requerirá nada más que papel y boli (y en este caso, su equivalente digital, el procesador de textos).

Más adelante, necesitaremos una maquina con IDE en VirtualBox

Y probar el código fuente con el que trabajaremos.

```
public class ClasificadorEdades {


    /**
     * Este método recibe una edad en años y devuelve:
     * - "ESTUDIAR" si edad está entre 0 y 18 años (inclusive)
     * - "TRABAJAR" si edad está entre 19 y 67 años (inclusive)
     * - "JUBILARSE" si edad es 68 o superior
     */
    public String clasificarEdad(int edad) {
        if (edad >= 0 && edad <= 18) {
            return "ESTUDIAR";
        } else if (edad >= 19 && edad <= 67) {
            return "TRABAJAR";
        } else {
            return "JUBILARSE";
        }
    }
}
```

1. Grafo de flujo y complejidad ciclomática

Pistas.

Ejemplo Complejidad ciclomática - Junco TIC

Ejemplo Grafo y Complejidad Ciclomática - youtube

Complejidad ciclomática – Wikipedia

Teoría sobre Deseño e realización de probas - MediaWiki

1.1 Grafo de flujo

Dibuja el grafo de flujo del método clasificarEdad.

Solución:

[Nodo 1: Inicio]

↓

[Nodo 2: if (edad >= 0 && edad <= 18)] → Sí → [Nodo 3: return "ESTUDIAR"]

↓ No

[Nodo 4: if (edad >= 19 && edad <= 67)] → Sí → [Nodo 5: return "TRABAJAR"]

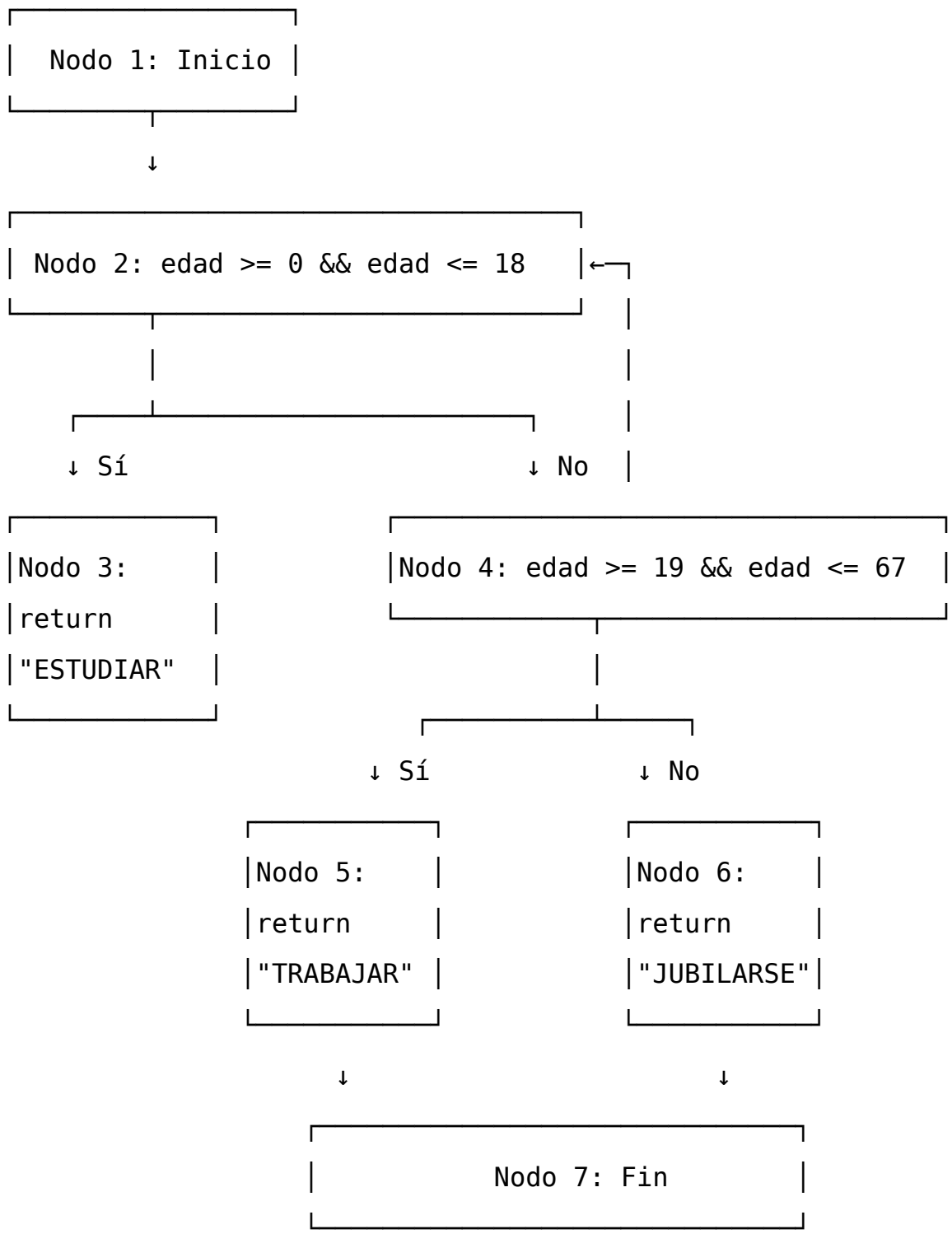
↓ No

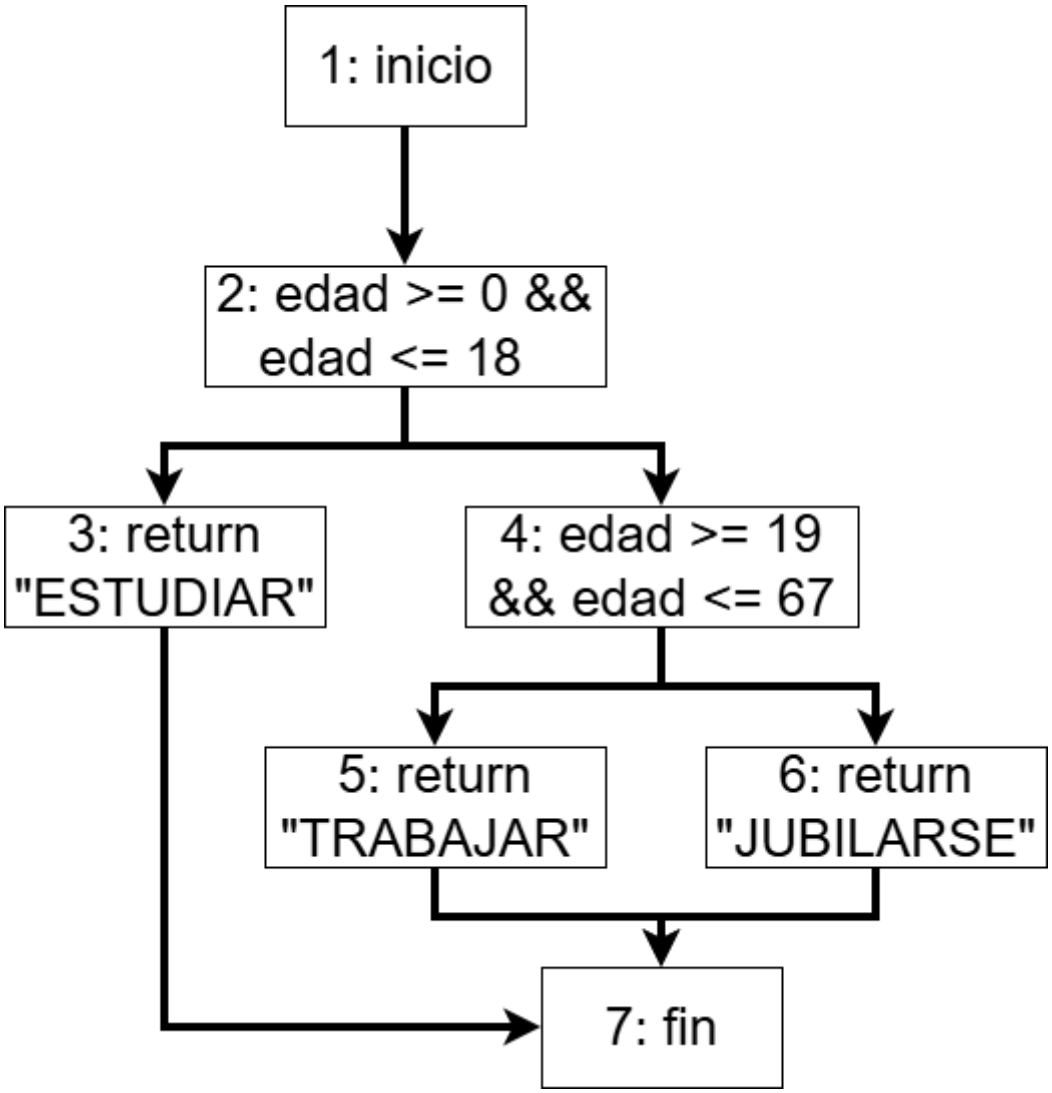
[Nodo 6: return ""JUBILARSE"]

↓

[Nodo 7: Fin]

En formato ascii:




1: inicio
2: edad >= 0 && edad <= 18
3: return "ESTUDIAR"
4: edad >= 19 && edad <= 67
5: return "TRABAJAR"
6: return "JUBILARSE"
7: fin

1.2. Complejidad ciclomática

Calcula la complejidad ciclomática usando la fórmula de McCabe.

Solución:

Número de arcos (E) = 7

Número de nodos (N) = 6

Complejidad = E - N + 2 = 7 - 6 + 2 = 3

También se puede calcular por regiones: 3 regiones cerradas → Complejidad = 3

2. Clases de Equivalencia

Pistas.

ISTQB CTFL v3.1. Cap 4 - Partición de Equivalencia - TestingBaires

Determinar las clases de equivalencia

Teoría sobre Deseño e realización de probas - MediaWiki

Clases de equivalencia y valores límite – ejemplo con edad – youtube.

Crea una tabla de Clases de Equivalencias.

Solución:

Entrada	**Clases Válidas**	**Clases No Válidas**
int edad	(1) edad >= 0 && edad <= 18	(No aplica)
	(2) edad >= 18 && edad <= 67	
	(3) edad > 67	

3. Análisis de Valores Límite (AVL)

Pistas.

Análisis de valores límite

Teoría sobre Deseño e realización de probas - MediaWiki

Técnica de Análisis de Valores Límites |- ejemplo velocidad - youtube

Propón al menos 3 casos de prueba usando valores límite.

Solución:

AVL1: n = 0 (valor más pequeño)

AVL2: n = 1 (valor más pequeño + 1)

AVL3: n = 17 (valor límite de edad - 1)

AVL4: n = 18 (valor límite de edad)

AVL5: n = 19 (valor límite de edad + 1)

AVL6: n = 66 (valor límite de jubilación - 1)

AVL7: n = 67 (valor límite de jubilación)

AVL8: n = 68 (valor límite de jubilación + 1)

4. Conjetura de errores

Pistas.

Conjetura de errores

Propón dos casos de prueba adicionales basados en posibles errores comunes.

Solución:

CE1: edad = Integer.NEGATIVE_VALUE (-1) - valor negativo

CE2: edad = Integer.MAX_VALUE (130) - valor demasiado alto

5. Tabla de Casos de Prueba

Pistas.

Casos de uso, resultados esperados y análisis.

Completa una tabla de Casos de Prueba.

Solución:

Caso	Entrada (edad)	Clase/AVL/CE	Salida Esperada
CP1	5	(1) Clase edad >= 0 && edad <= 18	"ESTUDIAR"
CP2	36	(2) Clase edad > 18 && edad <= 67	"TRABAJAR"
CP3	-3	(3) Clase edad > 67	"JUBILARSE"
CP4	18	AVL4	"TRABAJAR"
CP5	67	AVL7	"TRABAJAR"
CP6	-1	CE1	"ERROR"
CP7	130	CE2	"ERROR"