



# Pruebas Estructurales

Principios de Desarrollo de Software – Tema 5




1



## Agenda

- Definición y aplicabilidad de las pruebas de caja blanca
- Prueba del Flujo de Control
  - Prueba del Camino Básico
  - Prueba de las Estructuras de Control
- Ejercicios



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

2

## Definición

- Las pruebas de caja blanca permiten la prueba del elemento a probar estudiando la estructura interna del **código fuente** que implementa este elemento.
- El procedimiento general de las pruebas de caja blanca:
  - Se analiza la **estructura interna** de software que se pretende probar
  - Se identifican los distintos **caminos** de ejecución del software bajo prueba
  - Se identifican las **entradas** para que el software bajo prueba ejecute los caminos de prueba seleccionados
  - Se **ejecutan** las pruebas
  - Se **comparan** las salidas **obtenidas** con las salidas **esperadas**
  - Se determina si el software bajo prueba funciona tal y como se esperaba



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

3

3

## Aplicabilidad

- Generalmente, las pruebas de caja blanca se aplican durante las **pruebas unitarias** y son realizadas por los desarrolladores
- Se puede aplicar durante las pruebas de integración y sistema pero sirve para identificar distintas situaciones de colaboración entre clases más que la verificación del código de un método



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

4

4

## Desventajas

- El número de caminos de ejecución puede ser tan grande que no todos ellos pueden ser probados
- Los casos de prueba que se detecten puede que no sirvan para identificar errores propios de los datos
  - Por ejemplo:  $p=q/r$
- Las pruebas de caja blanca asumen que el flujo de control es correcto (o casi). Es bastante complicado identificar los errores que se deban a caminos de ejecución que no existen
- El probador debe tener habilidades de programación para entender y evaluar el software bajo prueba



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

5

5

## Ventajas

- Cuando se utilizan las pruebas estructurales, el probador puede asegurar que cada camino de ejecución del software bajo prueba se ha identificado y probado



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

6

6

## Técnicas de Pruebas Estructurales

- Pruebas del flujo de control
  - Identifica los caminos de ejecución de una unidad de software e identifica los casos de prueba necesarios para que estos caminos se ejecuten
    - Prueba del camino básico
    - Prueba de las estructuras de control
- Prueba del flujo de datos
  - Es similar a la prueba del flujo de control, pero está enfocada en la prueba de la definición, uso y destrucción de cada una de las variables y atributos de una clase o método
    - Prueba de flujo de datos estática
    - Prueba de flujo de datos dinámica



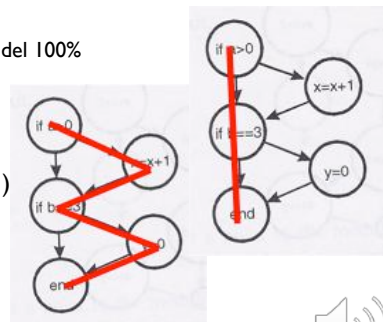
Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

7

7

## Niveles de cobertura - I

- En este contexto, es el porcentaje de código que se ha probado vs. código implementado
- En el flujo de control, la cobertura se puede definir como una serie de niveles diferentes
  - Nivel 0
    - Hay un nivel inferior a la cobertura del 100% de la secuencias
  - Cobertura de Secuencias (Nivel 1)
    - Consiste en generar los casos de prueba necesarios para pasar al menos una vez por cada secuencia



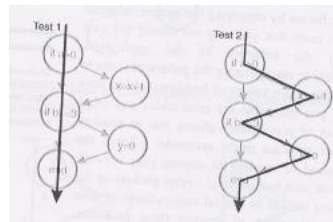
Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

8

8

## Niveles de Cobertura -2

- En el flujo de control, la cobertura se puede definir como una serie de niveles diferentes
  - Cobertura de decisiones (Nivel 2)
    - Consiste en generar el número suficiente de casos de prueba para asegurar que cada decisión en el código fuente tiene al menos una evaluación positiva (VERDADERO) y una evaluación negativa (FALSO)
    - Normalmente, la ejecución de las pruebas para la cobertura de decisión cumple con las restricciones establecida para la cobertura de secuencias



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

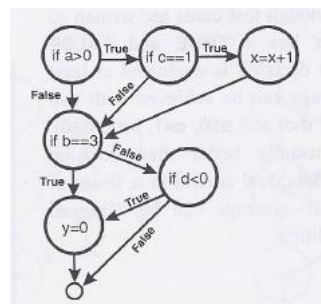


9

9

## Niveles de Cobertura -3

- En el flujo de control, la cobertura se puede definir como una serie de niveles diferentes
  - Cobertura de Condiciones (Nivel 3).
    - Consiste en la generación de suficientes casos de prueba para evaluar todas las posibles combinaciones de las decisiones establecidas en cada uno de los puntos de decisión



```
if (a > 0 && c == 1) {x = x + 1;}
if (b == 3 || d < 0) {y = 0;}
// note: || means logical OR
```

```
a > 0, c = 1, b = 3, d < 0
a ≤ 0, c = 1, b = 3, d ≥ 0
a > 0, c ≠ 1, b ≠ 3, d < 0
a ≤ 0, c ≠ 1, b ≠ 3, d ≥ 0
```



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

10

10

## Ejemplo

```
public class Caesar
{
    char chr;
    String map="";
    int inByte;
    final String
ALPHA="0123456789<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    public String encrypt(String x, int key)
    {
        String temp;
        this.setMap(key);
        x = x.trim(); //remove whitespace
        x = x.toLowerCase();
        for(int i = 0; i < x.length(); i++)
        {
            chr = x.charAt(i);
            temp+=map.charAt(ALPHA.indexOf(chr));
        }
        return temp;
    }
    public void setMap(int Key)
    {
        inByte=(inByte+ALPHA.length())%ALPHA.length();
        for (int i=0;i<ALPHA.length();i++)
            {map+=ALPHA.charAt((i+Key)%ALPHA.length());}
    }
}
```



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

11

11

## Prueba del camino básico – Definición

- Utiliza un análisis de la topología del grafo de control para identificar caminos de prueba
- La prueba de camino básico consiste en la ejecución de los siguientes pasos:
  - Derivar el grafo del flujo de control de un método a probar
  - Calcular la **complejidad ciclomática** del grafo
  - Seleccionar el valor obtenido como el número de caminos distintos a probar
  - Crear un caso de prueba para la ejecución de cada camino
  - Ejecutar estos casos de prueba



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

12

12

## Grafos del flujo de control

- Los grafos de flujo de control son la base de la prueba del flujo de control
- Estos grafos documentan la estructura de control del módulo
- Los módulos de código se convierten en gráficos, se analizan los caminos a lo largo del grafo y, a partir de ese análisis, se determinan los casos de prueba



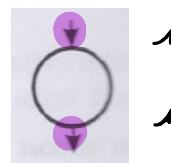
Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

13

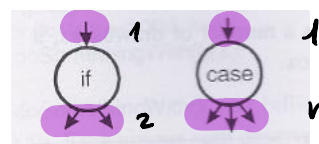
13

## Elementos del flujo de control

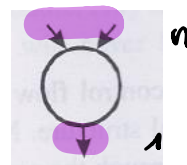
- Secuencia



- Punto de decisión



- Punto de unión



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

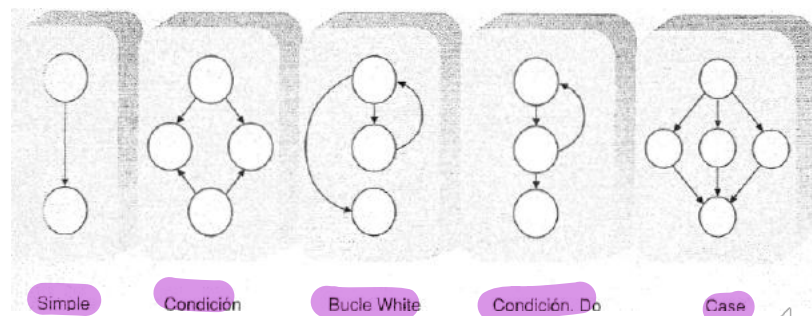
14

14



## Prueba del camino básico - I

- Los grafos del flujo de control **deben** **representar las distintas estructuras de control** de la siguiente manera:



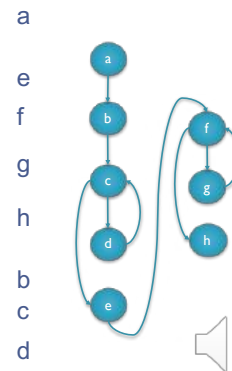
Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

15

15

## Ejemplo – Grafos de Flujo

```
public class Caesar
{
    char chr;
    String map="";
    int inByte;
    final String
    ALPHA="0123456789<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    public String encrypt(String x, int key)
    {
        String temp;
        this.setMap(key);
        x = x.trim(); //remove whitespace
        x = x.toLowerCase();
        for(int i = 0; i < x.length(); i++)
        {
            chr = x.charAt(i);
            temp+=map.charAt(ALPHA.indexOf(chr));
        }
        return temp;
    }
    public void setMap(int Key)
    {
        inByte=(inByte+ALPHA.length())%ALPHA.length();
        for (int i=0;i<ALPHA.length();i++)
        {map+=ALPHA.charAt((i+Key)%ALPHA.length());}
    }
}
```



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

16

16



## Prueba del camino básico -2

- El conjunto de caminos básicos de ejecución se pueden calcular utilizando la medida de la complejidad de McCabe
- Esto proporciona una estimación para asegurar cuántos casos de prueba se deben contemplar para garantizar una cobertura de decisiones
- La complejidad de McCabe se calcula de acuerdo con la siguiente fórmula:

$$V(G) = \text{Enlaces} - \text{Nodos} + 2$$

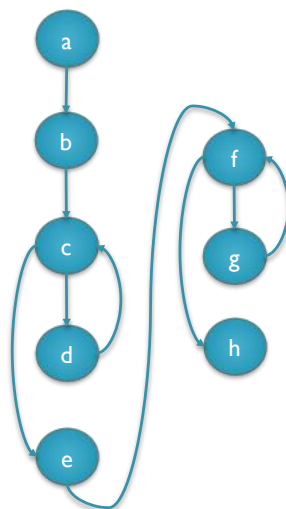


Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

17

17

## Ejemplo – Identificación de caminos básicos



$$V(G) = \text{Enlaces} - \text{Nodos} + 2$$

$$V(G) = 9 - 8 + 2 = 3$$



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

18

18

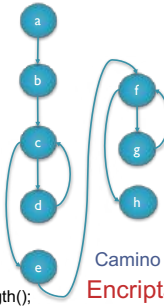
## Identificación de Caminos

```

public class Caesar
{
    char chr;
    String map="";
    int inByte;
    final String
    ALPHA="0123456789<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    public String encrypt(String x, int key)
    {
        String temp;
        this.setMap(key);
        x = x.trim(); //remove whitespace
        x = x.toLowerCase();
        for(int i = 0; i < x.length(); i++)
        {
            chr = x.charAt(i);
            temp+=map.charAt(ALPHA.indexOf(chr));
        }
        return temp;
    }
    public void setMap(int Key)
    {
        inByte=(inByte+ALPHA.length())%ALPHA.length();
        for (int i=0;i<ALPHA.length();i++)
        {map+=ALPHA.charAt((i+Key)%ALPHA.length());}
    }
}

```

$$V(G) = 9 - 8 + 2 = 3$$



Camino 1: a, b, c, e, ...

No posible

Camino 2: a, b, c, d {x}, e, f, h

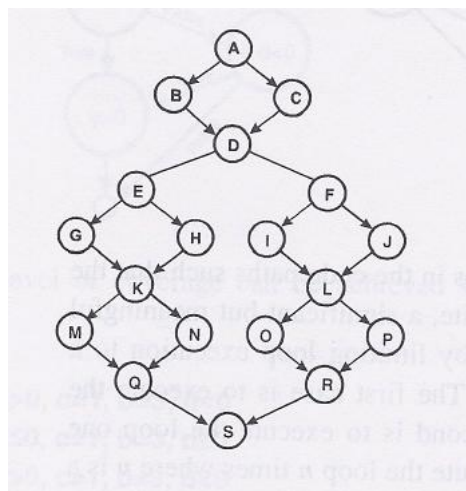
Encriptar cadena vacía

Camino 3: a, b, (c, d) {x}, e, (f, g) {x} h

Encriptar cadena NO va

## Prueba del camino básico – Ejemplo

-I

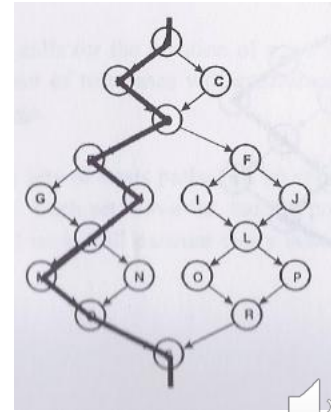
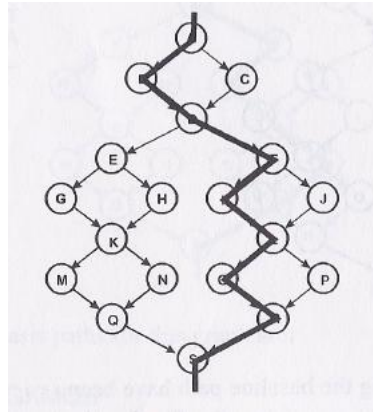


$$C = 24 - 19 + 2 = 7$$



## Prueba del camino básico – Ejemplo -3

- Determinar el conjunto de caminos básicos de acuerdo con las reglas anteriores



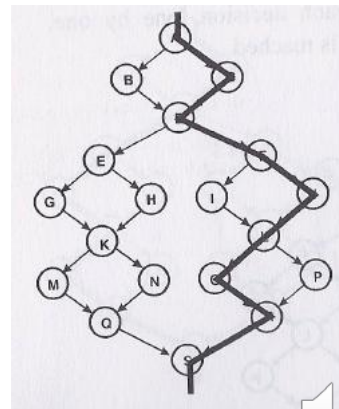
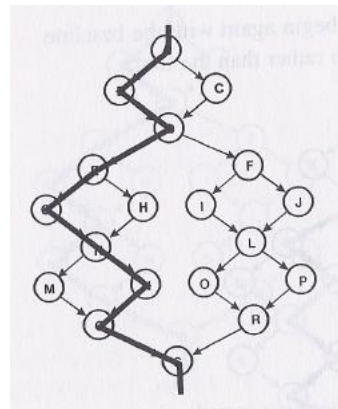
Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

23

23

## Prueba del camino básico – Ejemplo -4

- Determinar el conjunto de caminos básicos de acuerdo con las reglas anteriores



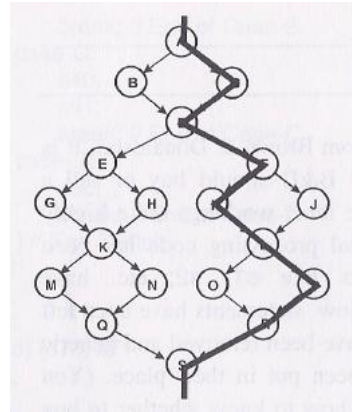
Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

24

24

## Prueba del camino básico – Ejemplo -5

- Determinar el conjunto de caminos básicos de acuerdo con las reglas anteriores



ABDEGKMQS  
ACDEGKMQS  
ABDFILORS  
ABDEHKMQS  
ABDEGKNQS  
ACDFJLORS  
ACDFILPRS



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

25

25

## Ejemplo Verify License

```
public boolean VerifyLicense(String LicenseFilePath) throws LMException {
    boolean result = false;
    // Step 1: try to read from the input JSON text file the different fields into a "License" object.
    License licenseToVerify = readLicenseFromJSON (LicenseFilePath);
    // Step 2: instantiate a new "LicensesStore" including all the generated licenses
    LicensesStore myStore = new LicensesStore ();
    // Step 3: Try to find the license obtained from the input file
    License licenseFound = myStore.Find(licenseToVerify);
    // Step 4: Check if the license found is valid at this moment
    if (licenseFound!=null){
        if (G result = licenseFound.IsValid();
    }
    return result;
}
```

$$V(G) = \text{Enlaces} - \text{Nodos} + 2$$

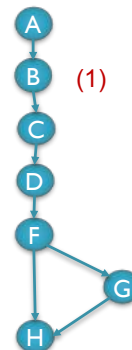
$$V(G) = 7 - 7 + 2 = 2$$

Camino 1: A, B, C, D, F, H

Licencia NO encontrada

Camino 2: A, B, C, D, g, H

Licencia encontrada



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

26

26



## Ejemplo Verify License (I)

```
private License readLicenseFromJSON(String path) throws LMException {
```

```
    License myLicense = null;
```

```
    String fileContents = "";
```

A.1

```
    BufferedReader reader;
```

A.2

```
    try {
```

```
        reader = new BufferedReader(new FileReader(path));
```

A.3

```
    } catch (FileNotFoundException e) {
```

```
        throw new LMException("Error: input file not found.");
```

A.4

```
    } 
```

A.5

```
    String line;
```

A.6

```
    try {
```

```
        while ((line = reader.readLine()) != null) {
```

```
            fileContents += line;
```

A.7

```
    } catch (IOException e) {
```

```
        throw new LMException("Error: input file could not be accessed.");
```

A.8

```
    } 
```

A.9

```
    try {
```

```
        reader.close();
```

```
    } catch (IOException e) {
```

```
        throw new LMException("Error: input file could not be closed.");
```

A.10

```
    } 
```

```
    // Transform the String with the file contents into a JSON object (in memory).
```

A.11

```
    JSONObject jsonLicense = null;
```

A.12

```
    try {
```

```
        jsonLicense = Json.createReader(new StringReader(fileContents)).readObject();
```

A.13

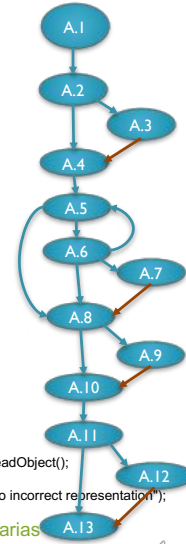
```
    } catch (JsonParseException ex) {
```

```
        throw new LMException("Error: JSON object cannot be created due to incorrect representation");
```

```
    } 
```

Caminos: Fichero No existe, Fichero Vacío, Fichero una o varias líneas, Fichero varias etiquetas JSON, Fichero JSON inválido  
Caminos no fáciles: Error mientras se lee el fichero, error para cerrar el fichero

Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales



$$V(G) = 14 - 1 = 13$$

$$V(G) = 18 - 13 + 2 = 7$$

27

## Ejemplo Verify License (I)

```
// Initialize license request
```

```
String stationName = jsonLicense.getString("Station Name");
```

```
String personInCharge = jsonLicense.getString("Person in charge");
```

```
String eMail = jsonLicense.getString("EMail");
```

```
String machineName = jsonLicense.getString("Machine Name");
```

```
String typeOfLicense = jsonLicense.getString("Type of License");
```

```
String licenseSignature = jsonLicense.getString("License");
```

A.13

```
myLicense = new License(stationName, personInCharge, eMail, machineName, typeOfLicense, licenseSignature);
```

```
// Check if all fields are valid and in range
```

```
checkLicenseFormat(myLicense);
```

(2)

```
return myLicense;
```

Caminos: Fichero con las etiquetas JSON correctas

Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales



28

28

## Ejemplo Verify License (2)

```
private void checkLicenseFormat(License lic) throws LMException {
```

```

A.13.1  if (lic.getPersonInCharge().length() < 1 || lic.getPersonInCharge().length() > 20) {
A.13.2      throw new LMException("Error: invalid String length for person in charge.");
A.13.3  }
A.13.4  // Length check for machine name
A.13.5  if (lic.getMachineName().length() < 1 || lic.getMachineName().length() > 10) {
A.13.6      throw new LMException("Error: invalid String length for machine name.");
A.13.7  }
A.13.8  // E-mail RFC822 compliant regex adapted for Java:
A.13.9  Pattern mailPattern = Pattern.compile("");
A.13.10 if (!mailPattern.matcher(lic.getEmail()).matches()) {
A.13.11     throw new LMException("Error: invalid E-mail data in JSON structure.");
A.13.12 }
A.13.13 // Length check for license request
A.13.14 if (!lic.getTypeOfLicense().equalsIgnoreCase("Fighter") || lic.getTypeOfLicense().equalsIgnoreCase("Starship") ||
A.13.15     lic.getTypeOfLicense().equalsIgnoreCase("All")) {
A.13.16     throw new LMException("Error: invalid type of license request.");
A.13.17 }
A.13.18 // Format check for license format (length and hex values)
A.13.19 if (lic.getSignature().length() != 64 || !lic.getSignature().matches("-?[0-9a-fA-F]*")) {
A.13.20     throw new LMException("Error: invalid String format for license key.");
A.13.21 }
A.13.22 }
```



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

29

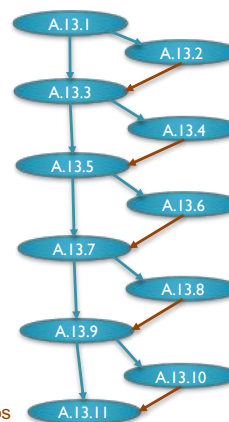
29

## Ejemplo Verify License (2)

```

private void checkLicenseFormat(License lic) throws LMException {
A.13.1  if (lic.getPersonInCharge().length() < 1 || lic.getPersonInCharge().length() > 20) {
A.13.2      throw new LMException("Error: invalid String length for person in charge.");
A.13.3  }
A.13.4  // Length check for machine name
A.13.5  if (lic.getMachineName().length() < 1 || lic.getMachineName().length() > 10) {
A.13.6      throw new LMException("Error: invalid String length for machine name.");
A.13.7  }
A.13.8  // E-mail RFC822 compliant regex adapted for Java:
A.13.9  Pattern mailPattern = Pattern.compile("");
A.13.10 if (!mailPattern.matcher(lic.getEmail()).matches()) {
A.13.11     throw new LMException("Error: invalid E-mail data in JSON structure.");
A.13.12 }
A.13.13 // Length check for license request
A.13.14 if (!lic.getTypeOfLicense().equalsIgnoreCase("Fighter") || lic.getTypeOfLicense().equalsIgnoreCase("Starship") ||
A.13.15     lic.getTypeOfLicense().equalsIgnoreCase("All")) {
A.13.16     throw new LMException("Error: invalid type of license request.");
A.13.17 }
A.13.18 // Format check for license format (length and hex values)
A.13.19 if (lic.getSignature().length() != 64 || !lic.getSignature().matches("-?[0-9a-fA-F]*")) {
A.13.20     throw new LMException("Error: invalid String format for license key.");
A.13.21 }
A.13.22 }
```

Caminos: Error Person in Charge, Error Machine Name, Error Email, Error Type of License, Error Signature, Todos los campos correctos



$$V(G) = 10 - 11 + 2 = 1$$

$$V(G) = 15 - 11 = 4$$



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

30

30



## Ejemplo Verify License

```

public boolean VerifyLicense(String LicenseFilePath) throws LMException {
    A boolean result = false;
    B // Step 1: try to read from the input JSON text file the different fields into a "License" object.
    C License licenseToVerify = readLicenseFromJSON (LicenseFilePath);
    D // Step 2: instantiate a new "LicensesStore" including all the generated licenses
    E LicensesStore myStore = new LicensesStore ();
    F // Step 3: Try to find the license obtained from the input file
    G License licenseFound = myStore.Find(licenseToVerify);
    H // Step 4: Check if the license found is valid at this moment
    I if (licenseFound!=null){
    J     result = licenseFound.IsValid();
    K }
    L return result;
}

```

$$V(G) = \text{Enlaces} - \text{Nodos} + 2$$

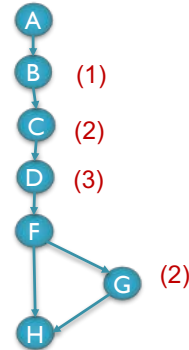
$$V(G) = 7 - 7 + 2 = 2$$

Camino 1: A, B, C, D, F, H

Licencia NO encontrada

Camino 2: A, B, C, D, G, H

Licencia encontrada



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

31

31

## Prueba de la estructura de control

- La prueba de camino básico solo proporciona una cobertura a nivel de decisión, pero esto no es suficiente en varios casos
- Otras variaciones incluyen la prueba de las estructuras de control:
  - Cobertura de condición
  - ✗ ◦ Prueba de bucles



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

32

32

## Cobertura de decisiones – Ejemplo -I

- Método a probar:

```
// Format check for license format (length and hex values)
if (lic.getSignature().length() != 64 || !lic.getSignature().matches("-?[0-9a-fA-F]+")) {
    throw new LMException("Error: invalid String format for license key.");
}
```

- Cobertura de Decisiones

- Tener un hash correcto (no entrar en la condición)
- Tener un hash incorrecto (entrar en la condición y lanzar la excepción)

- Cobertura de Condiciones

		lic.getSignature().matches("-?[0-9a-fA-F]+")	
		True	False
lic.getSignature().length() != 64	True	Excepción	Excepción
	False	Correcto	Excepción

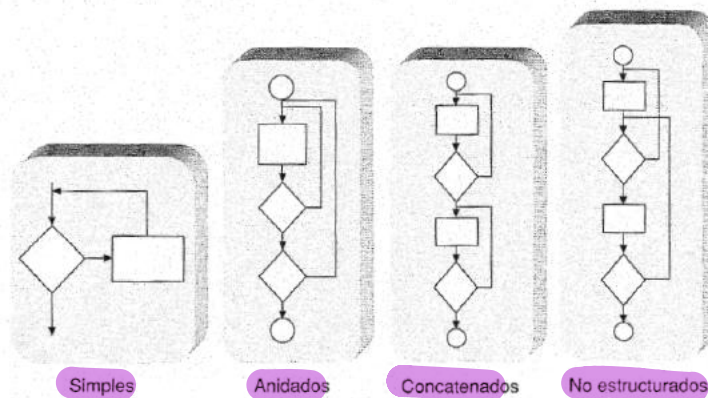


33

33

## Pruebas de bucle - I

- Tipos de bucle



No en java



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

34

34

## Pruebas de bucle -2

- **Simples:**
  - No pasar por el bucle
  - Pasar por el bucle 1 vez
  - Pasar por el bucles 2 veces
  - Pasar por el bucle max-1 veces
  - Pasar por el bucle el número máx. de veces
  - Intentar pasar por el bucle máx.+1 veces
- **Anidados:**
  - Comenzar en el bucle más interno y progresar hacia fuera
- **Concatenados:**
  - Considerarlos como bucles independientes
- **No estructurados**
  - Rediseñar los bucles. Problemas de Calidad



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

35

35

## Pruebas de Bucle - Ejemplo

```
private void Load () {
    try
    {
        JsonReader reader = new JsonReader(new
            FileReader(System.getProperty("user.dir") + "/Store/licensesStore.json"));
        Gson gson = new Gson();
        License [] myArray = gson.fromJson(reader, License[].class);
        this.licensesList = new ArrayList<License>();
        for (License lic: myArray) {
            this.licensesList.add(lic);
        }
    }
    catch (Exception ex)
    {
        this.licensesList = new ArrayList<License>();
    }
}
```

- No pasar por el bucle -> Fichero Vacío
- Pasar por el bucle 1 vez -> Fichero con 1 licencia
- Pasar por el bucles 2 veces -> Fichero con dos licencias
- Pasar por el bucle max-1 veces -> (i.e. Fichero con 99 licencias)
- Pasar por el bucle el número máx. de veces -> Fichero con 100 licencias
- Intentar pasar por el bucle máx.+1 veces -> No aplicable



Principios de Desarrollo de Software – Tema 9. Pruebas Estructurales

36

36