UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

Trabajo Teoría 2

Grupo C01
Ejercicio 3

Asignatura: Ingeniería del Software II

Grupo de Titulación (21/22): C

Titulación: Grado en Ingeniería Informática

Fecha: 22/12/2022

Ficha del Trabajo:

Equipo Nº: 3
Apellidos y Nombre
Víctor Ortega Gómez
Carlos Romero Navarro

1. Escribir, al menos el pseudocódigo correspondiente al método o a los métodos identificados

El código generado para resolver el problema se separa en dos clases, la clase AsignarCuenta, que contiene todo lo relacionado con el funcionamiento pedido por el problema, el método comprobar(), y la clase Cliente, que se realiza para simplificar la entrada de datos.

Así, el código correspondiente a estas clases es el siguiente:

```
public class AsignarCuenta {

   public static int edad = 18;
   public static boolean estudiante = true;
   public static boolean independizado = true;

   public static void main(String[]args) {

      Cliente cliente = new Cliente(edad, estudiante, independizado);

      String cuenta = comprobar(cliente);

      System.out.println(cuenta);
   }

   public static String comprobar(Cliente cliente) {

      int edad = cliente.getEdad();
      boolean estudiante = cliente.isEstudiante();
      boolean independizado = cliente.isIndependizado();

      if(edad < 18 && estudiante && !independizado)
         return "Cuenta: Comfort";

      else if(edad < 25 && estudiante && independizado)
         return "Cuenta: Vamos que tu puedes";

      else if(edad > 18 && edad < 25 && !estudiante && !independizado)
         return "Cuenta: Ahorra ahora que puedes";

      else if(edad > 18 && edad < 25 && !estudiante && independizado)
         return "Cuenta: Saltando del nido";

      else if(edad >= 25 && !estudiante && !independizado)
         return "Cuenta: Independizate que va siendo hora";

      else if(edad >= 25 && !estudiante && independizado)
         return "Cuenta: Bienvenido a la vida adulta";

      else return "No existe cuenta bancaria que se ajuste a ese perfil";
   }
```

```
}


public class Cliente {

   private int edad;
   private boolean estudiante;
   private boolean independizado;

   public Cliente(int edad, boolean estudiante, boolean independizado) {
      super();
      this.edad = edad;
      this.estudiante = estudiante;
      this.independizado = independizado;
   }

   public Cliente() {
      // TODO Auto-generated constructor stub
   }

   public int getEdad() {
      return edad;
   }

   public boolean isEstudiante() {
      return estudiante;
   }

   public boolean isIndependizado() {
      return independizado;
   }


}
```

2. Identificar las variables que se deben tener en cuenta para probar el método de interés.

Las variables para tener en cuenta en el método comprobar no son directamente sus parámetros, ya que este es un objeto de la clase Cliente. Las variables para tener en cuenta son el entero edad y los booleanos, estudiante e independizado. Estos cogen sus valores de los atributos del cliente introducido y trabajan con ellos en las condiciones propuestas.

3. Identificar los valores de pruebas para cada una de las variables anteriores usando las tres técnicas vistas en teoría, especificando para cada una cual es la que ha sido usada.

En la siguiente tabla podemos observar los distintos datos relacionados con el enunciado del ejercicio:

PARÁMETROS ENTRADA	CLASES DE EQUIVALENCIA	VALORES DE PRUEBA
edad	(-∞,0) U [0,18) U [18,25) U [25,∞)	{-10000000, -1, 0, 1, 17, 18, 19, 24, 25, 26, 1000000, NULL}
estudiante	{true, false}	{true, false, NULL}
independizado	{true, false}	{true, false, NULL}

Nota: El original de esta tabla y las vistas a lo largo del documento pertenecen al archivo Obtención de casos de prueba.xslx

El color rojo representa los valores correspondiente a la conjetura de errores; los valores azules a los valores límite en variante ligera; y, el color verde, representa los valores límite en su variante pesada.

4. Calcular el número máximo posible de casos de pruebas que se podrían generar a partir de los valores de pruebas (combinatoria).

El número máximo de casos de prueba será el producto cartesiano de todos los valores posibles que pueda tomar cada variable. Así, tenemos la siguiente fórmula:

n valores edad * n valores estudiante * n valores independizado = Max casos de pruebas

12*3*3 = 108 casos de prueba posibles a partir de este conjunto de valores de pruebas.

5. Defina un conjunto de casos de pruebas para cumplir con each use (cada valor una vez).

Usamos cada valor interesante al menos en un caso de prueba, obteniendo así tantos casos de prueba como valores se incluyan en la variable con más componentes. Se utilizará la estrategia "each use" vista en teoría para lograr esta cobertura.

Así, el conjunto de casos de prueba quedaría de la siguiente forma: (edad, estudiante, independizado)

- **CP1** (-10000000, false, false)
- **CP2** (-1, false, true)
- **CP3** (0, true, false)
- **CP4** (1, true, true)
- **CP5** (17, false, false)
- **CP6** (18, false, true)
- **CP7** (19, true, false)

- **CP8** (24, true, true)
- **CP9** (25, false, false)
- **CP10** (26, false, true)
- **CP11** (100000000, true, false)
- **CP12** (NULL, true, true)

6. Defina conjuntos de pruebas para alcanzar cobertura pairwaise usando el algoritmo explicado en clase. Se pueden comprobar los resultados con el programa PICT.

Tras comprobar los valores en la herramienta PICT (versión web: https://pairwise.yuuniworks.com/), se generan los siguientes casos de prueba incluidos en el anexo a este archivo, llamado *Resultados-pairwaise.txt.*

7. Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura de decisiones.

Para este apartado, se redirige al lector al archivo *Obtención de casos de prueba.xslx* (hoja 1), donde se podrá ver el proceso de obtención de casos de prueba para la cobertura de decisiones. Aquí podremos ver el conjunto final de casos de prueba tras la combinación de los casos de prueba repetidos.

PARÁMETROS ENTRADA	C1	C2	C3	C4	C5	C8	C9	C10	C12
edad	17	17	18	18	19	19	24	25	25
estudiante	false	true	true	true	false	false	true	false	false
independizado	true	false	false	true	false	true	true	false	true

8. Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura MC/DC.

Para este apartado, se redirige al lector al archivo *Obtención de casos de prueba.xslx* (hoja 2), donde se podrá ver el proceso de obtención de casos de prueba para la cobertura de decisiones. Aquí podremos ver el conjunto final de casos de prueba tras la combinación de los casos de prueba repetidos.

PARÁMETROS ENTRADA	C1	C2	C3	C4	C5	C6	C8	C9	C10	C11	C12
edad	20	17	26	18	17	19	19	24	25	24	25
estudiante	false	true	false	true	true	false	false	true	false	true	false
independizado	true	false	false	true	true	false	true	true	false	false	true

9. Comente los resultados del número de los casos de pruebas conseguidos en los apartados 4, 5 y 6 ¿qué podría decirse algo de la cobertura alcanzada?

En el apartado 4 se llega a la conclusión que tenemos un máximo de 108 posibles casos de prueba para el conjunto de valores propuestos. En el los apartados 5 y 6, mediante los criterios de combinación propuestos, podemos ver que en el 5, llegamos a tener 12 casos de prueba.
En el 6, como se ve en el archivo adjunto *Resultados-pairwaise.txt*, se consiguen un total de 37 casos de prueba. Si sumamos ambos, tendríamos 49 casos de 108 posibles. Este conjunto representaría un 45.37% de los casos posibles.

Sin embargo, creemos que la mejor cobertura sería each-use, ya que no se excede de casos de prueba como sería el pairwise ni recorre todos los posibles casos, ya que sería contraproducente.