Taller de preparación para la OCI - PUC

Control de flujo if - else: Operadores lógicos

1. Introducción

En este capítulo se tratarán los operadores lógicos **!** (not), **&&** (and) y || (or). Estos operadores tienen su origen en la lógica y permiten operar valores de verdad (o **boolean**s) para generar nuevos valores de verdad.

En el capítulo anterior vimos que las instrucciones **if** - **else if** - **else** nos permitían ejecutar ciertos bloques de código según se cumplían o no determinadas condiciones. Vimos también, que estas condiciones tenían que ser booleans y que resultaban de la comparación entre números o textos. Existe otra forma de obtener booleans y esta es a partir de la operación de o entre booleans.

Muchas veces no nos basta con una condición simple, sino que queremos alguna combinación de varias condiciones. Por ejemplo, en el caso de rendir el examen de conducir, el postulante debe ser mayor de 18 años **o** ser mayor de 17 **y** acreditar un curso en una escuela de conductores. También podemos tomar decisiones en base a condiciones que **no** se cumplen. Por ejemplo, si **no** tengo dinero suficiente para comprar una entrada al cine, me imagino la película.

2. Operadores lógicos

2.1. Negación - !

El primer operador lógico es la negación (**no** - **not**). Invierte el valor de verdad de un boolean: Si es `false` devuelve `true` y viceversa. Para usarlo se debe anteponer un signo de exclamación (**!**) al boolean que se desea negar. Por ejemplo (suponiendo que una entrada vale $3000):

```
1  int dinero = 2000;
2  boolean suficiente_dinero = dinero > 3000;
3  boolean insuficiente_dinero = !suficiente_dinero;
```

En este caso en la línea 2 se asigna a la variable `suficiente_dinero` el valor `false`, ya que `dinero` tiene asignado el valor 2000 y la comparación $2000 > 3000$ retorna `false`. En la línea 3 se asigna `true` a la variable `insuficiente_dinero`, ya que `suficiente_dinero` tiene asignado `false` y al anteponer el `!` se niega, transformándolo en `true`.

Es importante tener en cuenta que no solo se pueden negar variables, sino que cualquier expresión que sea booleana. En el caso anterior se podría haber hecho lo siguiente:

```
1  int dinero = 2000;
2  boolean insuficiente_dinero = !(dinero > 3000);
```

En este caso, la comparación $2000 > 3000$ retorna `false` y entonces, al anteponer `!` se obtiene el valor `true`. Finalmente se le asigna este valor (`true`) a la variable `insuficiente_dinero`.

2.2. Y - &&

El siguiente operador lógico es (**y** - **and**), también llamada conjunción. Este evalúa dos expresiones y nos dice si ambas son ciertas. Por ejemplo, puedo comprar una entrada para ver una película si:

1. quedan asientos desocupados y
2. tengo dinero suficiente.

Si alguna de las condiciones no se cumple, entonces no puedo comprar la entrada.

Formalmente, este operador evalúa el valor de verdad de dos booleans: si ambos son `true` devuelve `true` y en cualquier otro caso retorna `false`. El cuadro 1 muestra el resultado de evaluar `&&` con dos variables `a` y `b`.

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

Cuadro 1: a &&b

Por ejemplo (suponiendo que la sala tiene 40 asientos):

```
1 int asientos_vendidos = 20;
2 int dinero = 4000;
3 boolean posible_comprar = (asientos_vendidos < 40) && (dinero > 3000)
```

En el ejemplo anterior `(asientos_vendidos <40)` se evalúa como `true` y `(dinero >3000)` también como `true`. Luego `true && true` es `true` y ese es el valor que se asigna a la variable `posible_comprar`.

2.3. O - ||

El siguiente operador lógico es (**o** - **or**), también llamado conjunción. Este evalúa dos expresiones y nos dice si alguna de las 2 son ciertas (o ambas). Por ejemplo, iré al cine si:

1. están dando Mi Villano Favorito 3 o
2. están dando Mujer Maravilla

Si cualquiera de las condiciones se cumple, entonces voy a ir al cine.

Formalmente, este operador evalúa el valor de verdad de dos booleans: si alguno es `true` devuelve `true` y ambos son `false` devuelve `false`. El cuadro 2 muestra el resultado de evaluar || con dos variables `a` y `b`.

a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

Cuadro 2: a || b

Por ejemplo:

```
String cartelera = "Mujer Maravilla";
boolean voy_al_cine = cartelera.equals("Mi Villano Favorito 3") ||
                      cartelera.equals("Mujer Maravilla");
```

En el ejemplo anterior `cartelera.equals("Mi Villano Favorito 3")` se evalúa como `false` y `cartelera.equals("Muje Maravilla")` se evalúa como `true`. Luego `false || true` es `true` y ese es el valor que se asigna a la variable `voy_al_cine`.

3. Ejemplos

Primero resolveremos el ejercicio del cine del capítulo anterior sin utilizar operadores lógicos, para después utilizarlos y así ver su aporte:
Este ejercicio propone representar la situación de la introducción. Haz un programa que reciba la edad de una persona y un *string* que representa si la persona viene en compañía de un adulto (puede tener valores "sí" o "no"). Su programa debe imprimir un mensaje que anuncie si la persona puede entrar o no al cine. La persona puede entrar al cine si es mayor de 17 años o bien si es mayor de 15 años y va en compañía de un adulto.

3.1. Sin operadores lógicos

```
public class Main {

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Ingrese su edad:");
    int edad = scanner.nextInt();
    scanner.nextLine();
    System.out.println("Viene junto a un adulto? (si/no)");
    String acompanado = scanner.nextLine();
    if (edad > 17){
      System.out.println("Puede entrar al cine");
    }
    else if (edad > 15) {
      if (acompanado.equals("si")) {
        System.out.println("Puede entrar al cine");
      }
      else {
        System.out.println("No puede entrar al cine");
      }
    } else {
      System.out.println("No puede entrar al cine");
    }
  }
}
```

3.2. Con operadores lógicos

```
1
2  public class Main {
3
4    public static void main(String[] args) {
5      Scanner scanner = new Scanner(System.in);
6      System.out.println("Ingrese su edad:");
7      int edad = scanner.nextInt();
8      scanner.nextLine();
9      System.out.println("Viene junto a un adulto? (si/no)");
10     String acompanado = scanner.nextLine();
11     if ((edad > 17) || ((edad > 15) && acompanado.equals("si"))) {
12       System.out.println("Puede entrar al cine");
13     } else {
14       System.out.println("No puede entrar al cine");
15     }
16   }
17 }
```

Vemos que en la línea 11 hay un if cuya condición es el resultado de aplicar muchos operadores lógicos. Los paréntesis nos permiten agrupar las expresiones. En este caso tenemos 2 expresiones que se combinan mediante el operador **or**:
a) `(edad >17)`

b) `(edad >15) && acompanado.equals("si")`

Basta con que a) o b) sea `true` para entrar al if. Para ver si a) es `true` se aplica la comparación `edad >17`. b) se descompone a su vez en 2 expresiones que se combinan mediante el operador **and**

c) `(edad >15)`

d) `acompanado.equals("si")`

Para que b) sea `true`, c) y d) deben serlo: la edad debe ser supeior a 15 y acompañado debe tener el valor "si".