Unidad 7: anexo

Concepto de identidad e igualdad

Concepto de igualdad e identidad

- En general diremos que:
  - dos elementos son iguales cuando:
    - los valores de sus propiedades observables son iguales.
  - dos elementos son idénticos cuando:
    - al modificar una propiedad observable en uno de ellos, se produce una modificación en la del otro y viceversa.
  - de lo anterior, se deduce que:
    - identidad implica igualdad, pero igualdad no implica identidad.
- En Java (y conceptualmente hablando), la identidad es un concepto propio de los tipos objeto. No existe la identidad de tipos primitivos.

Tipos inmutables I

- Un objeto inmutable es aquél al que no se le pueden variar sus propiedades una vez que se ha creado.
- La inmutabilidad implica que:
  - Las propiedades son fijadas por el constructor cuando el objeto se crea.
  - Los métodos *set* no existen o son innecesarios.
  - Si existen métodos que modifican las propiedades del objeto, el resultado es otra instancia del tipo que contiene los datos modificados.

Tipos inmutables II

- Los tipos envoltura son inmutables: Integer, Double, Boolean…
- Existen otros tipos inmutables como String, Math.
- Cuando se modifica un tipo inmutable, se crea un nuevo objeto de manera que el nuevo objeto resultante **no es *idéntico*** al original.
- Ejemplo: ¿cuántos objetos se crean en este código? ¿cuánto valen a y b?
  - Double a = 5235.43;
  - Double b = a;
  - a = a - 7;

Operador de identidad ==

- El operador de identidad **==** y devuelve un valor **true** si el objeto a la izquierda del operador es **idéntico** al de la derecha.
- Si se usa el operador **==** sobre tipos primitivos, éste devuelve **true** si ambos valores son **iguales** y false en caso contrario.

El método equals

- Para el tipo objeto, la igualdad se consulta mediante el método equals.
- Es un método que tienen todos los tipos objeto en Java.
- Devuelve *true* si el objeto que lo invoca es **igual** al que se le pasa como parámetro. En otro caso, devuelve *false*.
- Cumple:
  - Simetría: si un objeto es igual a otro, el segundo también es igual al primero.
  - Transitividad: si un objeto es igual a otro, y éste segundo es igual a un tercero, el primero también será igual al tercero.

```
...
boolean equals(Object o);
...
```

El método equals

- Tiene las siguientes propiedades:
  - Simetría: si un objeto es igual a otro, el segundo también es igual al primero.
  - Transitividad: si un objeto es igual a otro, y éste segundo es igual a un tercero, el primero también será igual al tercero.

```
                                ...
        boolean equals(Object o);
                                ...
```

Ejemplos de uso de = y == (I)

- Sobre tipos básicos:
  - int i = 7;
  - int j = 4;
  - int k = 4;
  - Boolean a = (i == j) ; // a es false
  - Boolean b = (k == j) ; // b es true

Ejemplos de uso de = y == (II)

- Sobre tipos objeto:
  - Punto p1 = new PuntoImpl(1.0,1.0);
  - Punto p2 = new PuntoImpl(1.0,1.0);
  - Boolean c = (p1 == p2) ; // c es false
  - Boolean d = p1.equals(p2); // d es true

Ejemplos de uso de = y == (III)

- Aplicación sobre objetos:
  - Punto p1 = new PuntoImpl(1.0,1.0);
  - Punto p2 = new PuntoImpl(3.0,1.0);
  - Punto p3 = p1;
  - p1.setX(3.0);
  - Boolean a = (p3 == p1); // a es true
  - Boolean b = (p3 == p2); // b es false
  - Boolean c = p3.equals(p2); // c es true

Ejemplos de uso de = y == (IV)

- ¿Cuál será el valor de e? ¿Y de f?
  - Integer a = 3235;
  - Integer b = a;
  - a++;
  - Boolean e = (a==b);
  - Boolean f = a.equals(b);