ELO329 - Diseño y Programación Orientados a Objetos

Clases Object, ArrayList y Class: Reutilización de código y código genérico

Agustín González
Patricio Olivares

Clase `Object`: Nivel máximo de la jerarquía de clases

-Toda clase en Java hereda de la clase Object (ver en documentación) en su jerarquía máxima.

- Esta **no requiere ser indicada en forma explícita**.
- Esto permite que podamos agrupar en forma genérica elementos de cualquier clase, por ejemplo en un arreglo de Object.
- En esta clase hay métodos como equals() y toString() que en la mayoría de los casos conviene redefinir. ver documentación de clase Object.
- Ver: `EqualsTest.java`

Programación Genérica

- Las facilidades que ofrece el diseño orientado a objetos y la programación orientada a objetos permiten crear soluciones genéricas.
- La idea es crear código útil para situaciones similares.
- Por ejemplo podemos definir una clase con métodos como:

```
static int find (Object [ ] a , Object key) {
    int i;
    for (i=0; i < a.length; i++)
        if (a[i].equals(key)) return i; //éxito
    return -1;
    // no exitoso
}
```

- Este método funciona con cualquier arreglo mientras la clase de sus elementos tenga bien definido el método `equals`.

ArrayList: como Ejemplo de programación genérica

- Hay muchas estructuras de datos que no quisiéramos programar cada vez, ejemplo: `stack`, `lista`, etc.
- El `ArrayList` permite crear arreglos de tamaño variable (ver ArrayList en documentación) .
- Lo malo es que el acceso **no** es con `[ ]`.
- Ver ejemplo `CatsAndDogsV3.java`.

La clase `Class`

- La máquina virtual Java mantiene información sobre la estructura de cada clase. Esta puede ser consultada en tiempo de ejecución.

```
Employee e = new Employee(...);
...
Class cl=e.getClass();
```

- La instancia de `Class` nos sirve para consultar datos sobre la clase, por ejemplo, su nombre.

```
System.out.println(e.getClass().getName()+" "+e.getName()); // Genera Employee Harry Hacker
```

La clase `Class` (cont.)

- Ver la clase Class. Nos permite obtener toda la información de una clase, su clase base, sus constructores, sus campos datos, métodos, etc.
- Por ejemplo ver `ReflectionTest.java`
- Esta funcionalidad normalmente es requerida por constructores de herramientas más que por desarrolladores de aplicaciones.