La Classe Object

La classe Object del paquet **java.lang** és una classe especial de la qual hereten, directament o indirectament, totes les classes de Java. És la superclasse per excel·lència, ja que se situa en la cúspide de l'estructura d'herències entre classes.

Totes les classes que componen la API descendeixen de la classe Object. Fins i tot qualsevol classe que implementem nosaltres hereta de Object. Aquesta herència es realitza per defecte, sense necessitat d'especificar res. Per exemple, la definició de la classe Persona.

```
class Persona{
...
}
```

és en realitat, equivalent a:

```
class Persona extends Object{
...
}
```

I qualsevol classe que herete de Persona està heretant, al seu torn, de Object.

Quin és l'objectiu que totes les classes hereten de Object?
Fent això s'aconsegueix:

- Que totes les classes implementen un conjunt de mètodes – en Object només s'han definit mètodes – que són d'ús universal a Java, com realitzar comparacions entre objectes, etc. La funció d'aquests mètodes és ser re-implementats a la mesura de cada classe.

- Es pot referenciar qualsevol objecte, de qualsevol tipus, mitjançant una variable de tipus Object.
  Si volem veure els mètodes de Object que ha heretat Persona, escriurem Persona, seguida d'un punt(.) es desplegaran tots els atributs i mètodes disponibles: els propis, més els heretats de Object.

Vegem els mètodes més importants de Object, heretats per totes les classes de Java.

1. Mètode toString()

Aquest mètode està pensat perquè retorne una cadena que representa l'objecte que l'invoca amb tota la informació que interesse mostrar.

```
public String toString()
```

La seua implementació en la classe Object consisteix a retornar el nom qualificat de la classe a la qual pertany l'objecte, seguida d'una arrova (@) al costat de la referència de l'objecte. Per a un objecte Persona retorna una cosa similar a:

```
“paquete.Persona@2a139a55”
```

Aquesta implementació per defecte no és útil per a representar la majoria dels objectes, per la qual cosa ens veiem obligats a realitzar una sobreescriptura (overriding) de toString() en cada classe, que és on es troba la informació que volem representar.

Reimplementarem toString() en Persona; podem triar com volem representar una persona, però en aquest cas decidim que una representació adequada consisteix en el nom al costat de l'edat, ometent l'alçada.

```
Class Persona{
    …
    @Override
    public String toString(){
        String cad;
        cad = “Persona: “ + nombre + “ (“ + edad + “)”;
        return cad;
    }
}
```

Ha de declarar-se public, igual que en la classe Object, ja que tot mètode que substitueix a un altre ha de tindre, almenys el mateix nivell d'accés.

Ara podem mostrar per consola la informació d'un objecte Persona.

```
Persona p = new Persona(“Claudia”, 8, 1.20);
System.out.println(p.toString());
```

En realitat, System.out.println(p); //equival a System.out.println(p.toString());

2. Mètode equals()

Compara dos objectes i decideix si són iguals, retornant true en cas afirmatiu i false en cas contrari.

```
Persona a = new Persona(“Claudia”, 8, 1.20);
Persona b = new Persona(“Claudia”, 8, 1.20);
Persona c = new Persona(“Pepe”, 24, 1.89);
System.out.println(a.equals(b)); // true
System.out.println(a.equals(c)); // false
```

3. Mètode getClass()

És comú usar una variable Object per a referenciar un objecte de qualsevol classe que, com sabem, sempre serà una subclasse de Object. A vegades necessitem saber quina és aqueixa classe.

Per a això està el mètode getClass(), definit en Object i heretat per totes les classes. Aquest mètode, invocat per un objecte qualsevol, retorna la seua classe que, a la vegada, és un objecte de la classe Class. Totes les classes de Java, incloses Object i la pròpia Class, són objectes de la classe Class. Per exemple, si escrivim

```
Object a = “Luis”;
System.out.println(a.getClass()); // obtenim class java.lang.String
```

El mètode getName(), de la classe Class, retorna el nom qualificat de la classe invocant.

```
Object b = Double.valueof(3.5);
Class classe = b.getClass();
Class superclasse = classe.getSuperclass();    //superclasse: class java.lang.Number
System.out.println(superclasse.getName());     // nombre: java.lang.Number
```