2. Introducción a la POO

Julio Vega

julio.vega@urjc.es

Contenidos

- Hasta ahora habéis aprendido a programar de forma estructurada.
  - Pero hay más; eso solo han sido los primeros pasos en programación.
- Programación estructurada: forma habitual de programar (vs. *GoTo*).
  - Todo parte del *main()*, escribiendo las líneas en orden.
  - Existen funciones que encapsulan varias líneas y hacen *algo*.
  - Cuando hay que usar esas funciones, se invocan.
  - Cuando estas acaban, retornan al punto desde el que se les ha llamado.

- La programación nace para facilitarnos la vida.
- Con ella somos capaces de plasmar y resolver problemas reales.
- Pero si miramos el mundo real, ¡vemos objetos por todas partes!
  - Gente, animales, plantas, coches, aviones, edificios, ordenadores.
- En conclusión, los humanos pensamos en términos de objetos.
  - E.g. manejamos una silla; no un conjunto de patas y un respaldo.
- Dividimos los objetos en dos categorías: animados e inanimados.
  - Los animados se mueven, hacen cosas; los inanimados no por sí solos.
  - Pero todos tienen atributos y muestran un comportamiento.
    - E.g. atributos: tamaño, forma, color, peso.
    - E.g. comportamientos: una pelota rueda, rebota, se desinfla, etc.

- Los humanos aprendemos sobre los objetos existentes.
  - Estudiando sus atributos y observando sus comportamientos.
  - Distintos objetos pueden tener atributos y comportamientos similares.
- El diseño orientado a objetos modela el SW y lo acerca al mundo real.
  - Este diseño aprovecha la relaciones entre los objetos.
    - E.g. los objetos de la clase vehículo tienen las mismas caract.
  - Este DOO también aprovecha las relaciones de herencia.
    - Las nuevas clases se derivan de las existentes, refinándolas.
  - El DOO también modela la comunicación entre los objetos.
    - Los objetos se comunican, al igual que se comunican las personas.

- La idea del DOO es: objeto = atributos + operaciones.
- Además, los objetos pueden/deben ocultar información.
  - Los objetos deben poder comunicarse entre sí.
    - A través de interfaces bien definidas.
  - Pero no tienen que saber cómo están implementados otros objetos.
    - Los detalles de la implementación se ocultan dentro de los objetos.
- Podemos conducir un coche sin saber cómo funciona la transmisión.
  - Siempre que sepamos usar los pedales, el volante, etc.
- Ocultar información es imprescindible para un buen diseño software.
  - Ya lo veremos más adelante...

- Lenguajes como Java o C++ son orientados a objetos.
- La programación en estos lenguajes se llama POO.
  - Permiten a los programadores implementar un DOO.
- Otros lenguajes, como C, son por procedimientos.
- La unidad en C es la función; en C++, la clase.
  - Y a partir de la clase se instancian (crean) los objetos.
- Las clases contienen funciones que implementan operaciones.
  - Y datos que implementan atributos.

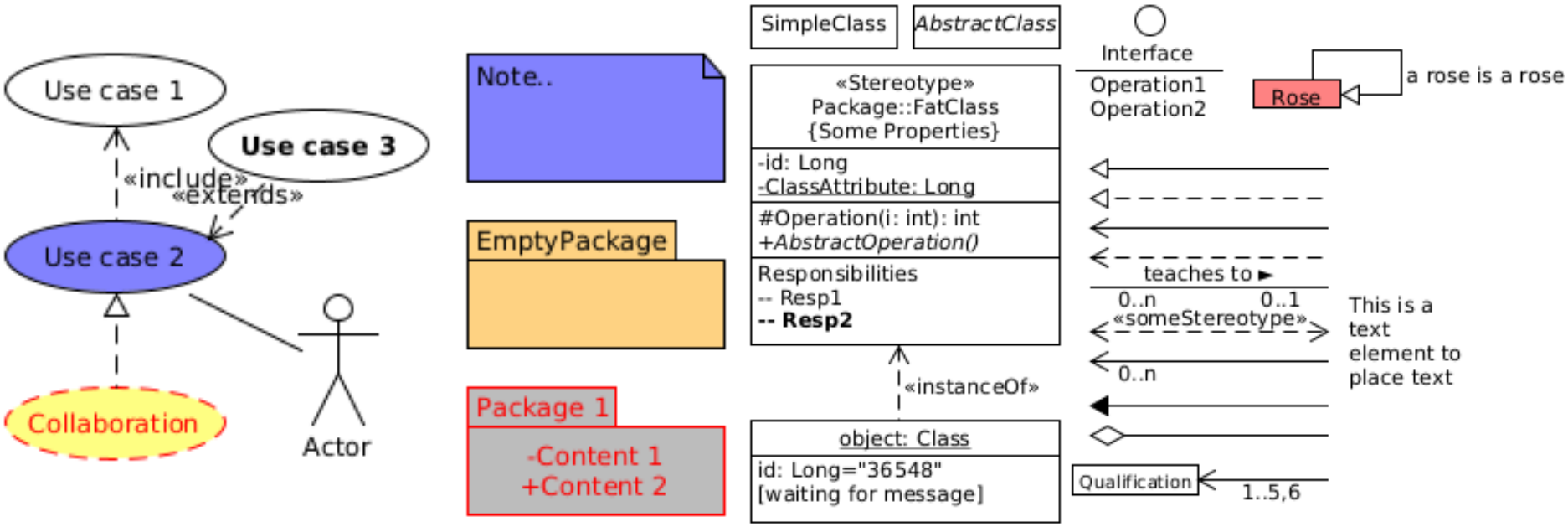
- Una clase es un tipo (enriquecido) definido por el usuario.
  - Contiene datos y también las funciones que manipulan estos.
  - También proporciona servicios a clientes (clases que usan clases).
  - Atributos: componentes de datos de la clase (miembros de datos).
  - Métodos: componentes de funciones de la clase (funciones miembro).
- Las clases son a los objetos lo que los planos de obra para las casas.
  - Una clase es un plano para construir un objeto de esa clase.
  - (O un molde con el que obtener figuras con esa forma.)
  - Podemos obtener (instanciar) varios objetos de una clase.
- Las clases pueden tener relaciones (asociaciones) con otras clases.
  - Es una bondad de la POO, la reutilización (de clases).
  - En DS las clases relacionadas se empaquetan en componentes.
  - (Un mantra del DS es reutilización, reutilización, reutilización.)

- *Picar código* (teclear sin más) puede valer para pequeños programas.
- Trabajar en equipo y/o en un gran sistema sw requiere mucho más.
  - Seguir *proceso* detallado para **analizar** los requerimientos del sistema.
    - *Qué* es lo que se supone que debe hacer el sistema.
  - Desarrollar un **diseño** que cumpla con esos requerimientos.
    - *Cómo* debe hacerlo el sistema.
- Puede implicar analizar y diseñar el sistema desde punto de vista OO.
  - Se denomina proceso de análisis y diseño orientado a objetos (A/DOO).
- Os repetiré este mantra: *hacer un buen A/D ahorra mucho tiempo*.
  - Primero ten claro *qué* y *cómo* y luego finalmente *impleméntalo*.

- Cuando problemas y n.º programadores $\uparrow \Longrightarrow$ unificar proceso.
- $\exists$ muchos A/DOO $\Longrightarrow$ un lenguaje gráfico muy popular es UML.
  - Unified Modelling Language o Lenguaje Unificado de Modelado.
- Surge en los 90s, cuando muchas empresas comienzan a usar POO.
  - E.g. HP, IBM, Microsoft, Oracle y Rational Software.
    - Estas y otras se unieron como socias de UML (*UML Partners*).
  - Lo desarrolla la *Rational Software Corporation*[1] (ahora de IBM).
    - Gracias en gran medida a las proposiciones del *OMG.org*.
    - La *Object Management Group* es una org. sin ánimo de lucro.
    - Promueve la estandarización de las tecnologías orientadas a objetos.
- UML 1.1 (1997) es mantenido por OMG hasta ahora, con UML 2[2].

[1] *https://www.ibm.com/docs/en/rational-soft-arch/9.5?topic=diagrams-creating-uml-m*

[2] *https://www.uml.org*

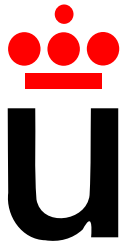


- Esquema de representación gráfica más usado para modelar sist. OO.
- Es un lenguaje gráfico complejo, con muchas características.
  - Nosotros usaremos un subconjunto conciso y simplificado de estas.
- Un analista puede diseñar sistemas usando varios procesos.
  - Pero todos se expresan mediante esta notación gráfica estándar.

2. Introducción a la POO

Julio Vega

julio.vega@urjc.es



Universidad
Rey Juan Carlos