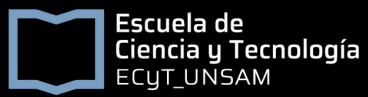
Argentina programa 4.0

Programación orientada a objetos

Clases - Métodos - Herencia

Universidad Nacional de San Martín

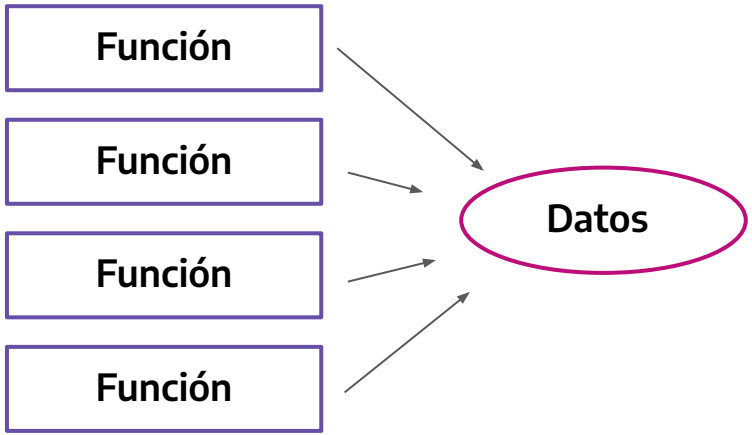


Escuela de Ciencia y Tecnología
ECyT_UNSAM

Secretaría de Economía del Conocimiento

Programación

Estructurada



Ej: Fortran, C, pascal

Limitaciones

- La descomposición de programas en funciones no es suficiente para conseguir la reusabilidad de código.
- Es difícil entender, mantener y actualizar un programa que consiste de una gran colección de funciones.

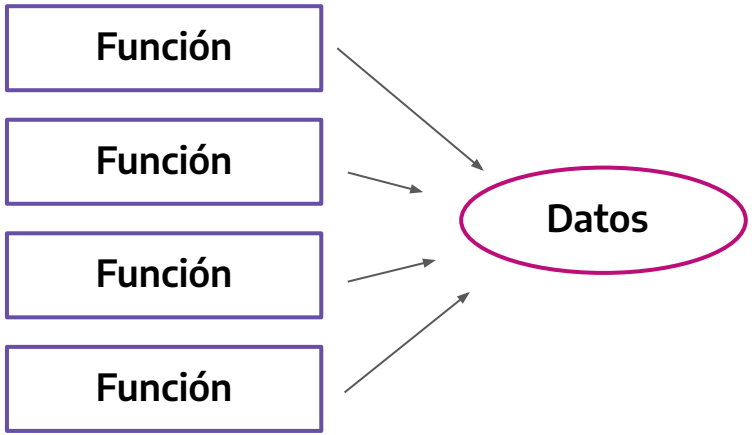
Solución:

Programación orientada a objetos

- Estilo de programación en el que las tareas se resuelven mediante la colaboración de objetos.

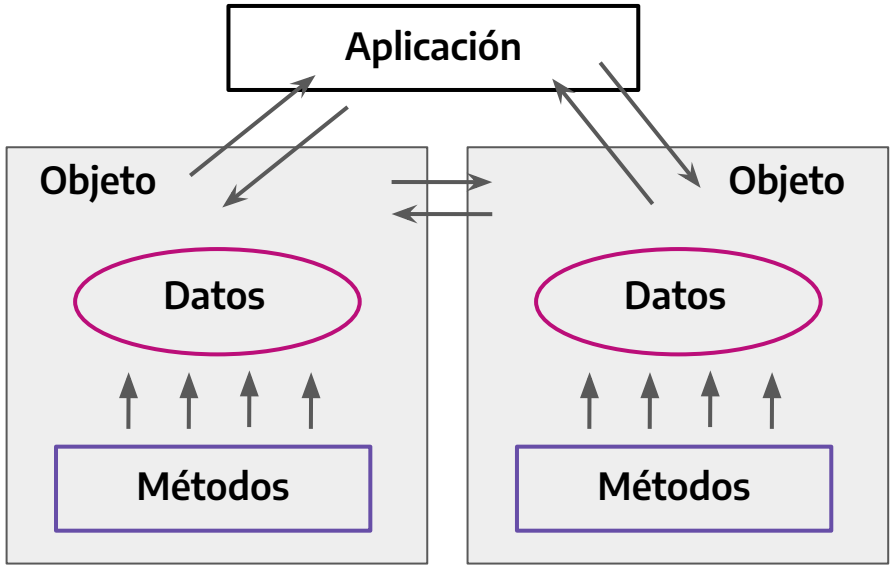
Programación

Estructurada



Ej: Fortran, C, pascal

Orientada a objetos (POO)



Ej: Java, C++, Python

Programación orientada a objetos

- **Clase:**
  - Es una plantilla o molde de entidades del mismo tipo
  - Una realización particular de una clase se llama instancia
  - Para crear una instancia se invoca el constructor

    ejemplo: nombre_instancia = nombre_clase(args)

- **Objeto:**
  - Es una instancia particular de una clase
  - Formada por: atributos de datos (variables) y comportamiento (métodos)
  - Para acceder un atributo/método se usa el operador punto

    ejemplo: nombre_objeto.nombre_atributo

    nombre_objeto.nombre_método

POO - Un ejemplo

- **Creamos una clase llamada Mascota**
  - Dentro de la clase vamos a definir los atributos

    ```
    Nombre
    Especie
    Peso
    ```

    Atención: Cuando creamos una nueva mascota, necesitamos dar estas informaciones (atributos de datos)
  - También dentro de clase vamos a definir los métodos: funciones que nos van a permitir interactuar con nuestros datos

    ```
    PuedeCorrer()
    Identificacion()
    ```

- **Una vez definida la clase, la vamos a usar:**
  - Creamos una instancia:

    ```
    tito = Mascota("Tito","Perro",6)
    ```
  - Le aplicamos un método a la instancia creada:

    ```
    tito.Identificacion()
    ```

Otro ejemplo:

- Queremos hacer un programa de control de cuentas de clientes en una red bancaria
- Necesitamos guardar las informaciones sobre cada cuenta bancaria
- Creamos una clase para representar exactamente este tipo de dato cuenta que agregue las informaciones y comportamientos básicos de una cuenta corriente.

- Creamos la clase Cuenta
  - Atributos:

    ```
    saldo
    numero
    ```

  - Métodos:

    ```
    resumen()           (imprime CC número numero tiene saldo  saldo)
    extraccion(valor)   (sustrae el valor dado a saldo)
    deposito(valor)     (adiciona el valor dado a saldo)
    ```

Otro ejemplo:

- **Creamos la clase Cuenta**
  - Atributos:

```
saldo
numero
```

  - Métodos:

```
resumen()              (imprime CC número numero tiene saldo  saldo)
extraccion(valor)  (sustrae el valor dado a saldo)
deposito(valor)    (adiciona el valor dado a saldo)
```

- **Creamos el objeto c1 con CC 1234-5 y saldo de $100**

```
[]   c1 = Cuenta (1234-5, 100)
```

- **Le pedimos el saldo en cuenta**

```
[]   c1.resumen()
```

Otro ejemplo:

- **Creamos la clase Cuenta**
  - Atributos:

```
saldo
numero
```

  - Métodos:

```
resumen()              (imprime CC número numero tiene saldo  saldo)
extraccion(valor)  (sustrae el valor dado a saldo)
deposito(valor)    (adiciona el valor dado a saldo)
```

- **Creamos el objeto c1 con CC 1234-5 y saldo de $100**

```
[]   c1 = Cuenta (1234-5, 100)
```

- **Le pedimos el saldo en cuenta**

```
[]   c1.resumen()
```

CC número `1234-5` tiene saldo `100`

Otro ejemplo:

- **Creamos la clase Cuenta**
  - Atributos:

```
saldo
numero
```

  - Métodos:

```
resumen()              (imprime CC número numero tiene saldo  saldo)
extraccion(valor)  (sustrae el valor dado a saldo)
deposito(valor)    (adiciona el valor dado a saldo)
```

- **Creamos el objeto c1 con CC 1234-5 y saldo de $100**

```
[]   c1 = Cuenta (1234-5, 100)
```

- **Le pedimos el saldo en cuenta**

```
[]   c1.resumen()
```

- **Retiramos de la cuenta $30**

```
[]   c1.extraccion(30)
```

Otro ejemplo:

- **Creamos la clase Cuenta**
  - Atributos:

```
saldo
numero
```

  - Métodos:

```
resumen()              (imprime CC número numero tiene saldo  saldo)
extraccion(valor)  (sustrae el valor dado a saldo)
deposito(valor)    (adiciona el valor dado a saldo)
```

- **Creamos el objeto c1 con CC 1234-5 y saldo de $100**

```
[]   c1 = Cuenta (1234-5, 100)
```

- **Le pedimos el saldo en cuenta**

```
[]   c1.resumen()
```

- **Retiramos de la cuenta $30**

```
[]   c1.extraccion(30)
```

- **Le pedimos nuevamente el saldo en cuenta**

```
[]   c1.resumen()
```

Otro ejemplo:

- **Creamos la clase Cuenta**
  - Atributos:
    ```
    saldo
    numero
    ```
  - Métodos:
    ```
    resumen()              (imprime CC número numero tiene saldo  saldo)
    extraccion(valor)  (sustrae el valor dado a saldo)
    deposito(valor)     (adiciona el valor dado a saldo)
    ```

- **Creamos el objeto c1 con CC 1234-5 y saldo de $100**

```
[]   c1 = Cuenta (1234-5, 100)
```

- **Le pedimos el saldo en cuenta**

```
[]   c1.resumen()
```

- **Retiramos de la cuenta $30**

```
[]   c1.extraccion(30)
```

- **Le pedimos nuevamente el saldo en cuenta**

```
[]   c1.resumen()
```

CC número `1234-5` tiene saldo `70`

Principio de la POO

- Usamos clases y objetos para facilitar la construcción de programas más complejos
- Notemos que en los métodos de `extraccion` y `deposito` no es necesario pasar el saldo como parámetro, solo pasamos el valor a ser sacado o depositado
- Este efecto de memoria o de permanencia de los atributos es intencionado para evitar estar pasando muchos parámetros cuando estamos lidiando con informaciones complejas en nuestros programas.
- La idea es imitar el comportamiento de objetos del mundo real: al hacer un depósito o extracción en una cuenta, no necesitamos informar al banco cuál es el saldo de la cuenta.
- Una clase puede ser instanciada varias veces: o sea podemos tener muchos objetos, en nuestro ejemplo diferentes personas: c1, c2, c3, ... , cN (todos los clientes del banco)
- El programa en ejecución es como una tela de objetos que se comunican entre sí a través de mensajes