Programación Orientada a Objetos

Clase #17

IIC1103 – Introducción a la Programación

Marcos Sepúlveda (marcos@ing.puc.cl)

Veremos hoy …

- Interacción de clases

Interacción de clases

- Esta idea es central a la resolución de problemas mediante programación O-O:
  - Un programa O-O es una colección de clases que interactúan entre sí;
  - Cada clase ofrece ciertos servicios a las otras clases, pero a su vez requiere de servicios ofrecidos por otras clases.

- A la hora de la ejecución de un programa
  - Los servicios ofrecidos por los objetos (instancias) de una cierta clase son utilizados por objetos de otras clases.
  - Para acceder a los servicios, los objetos se comunican a través de sus métodos.

Interacción de clases

- Conceptos claves
  - **Atributos** de una clase pueden ser objetos de otras clases
  - **Métodos** de una clase pueden llamar a métodos de otras clases
  - Se pueden enviar **objetos** como **parámetros**

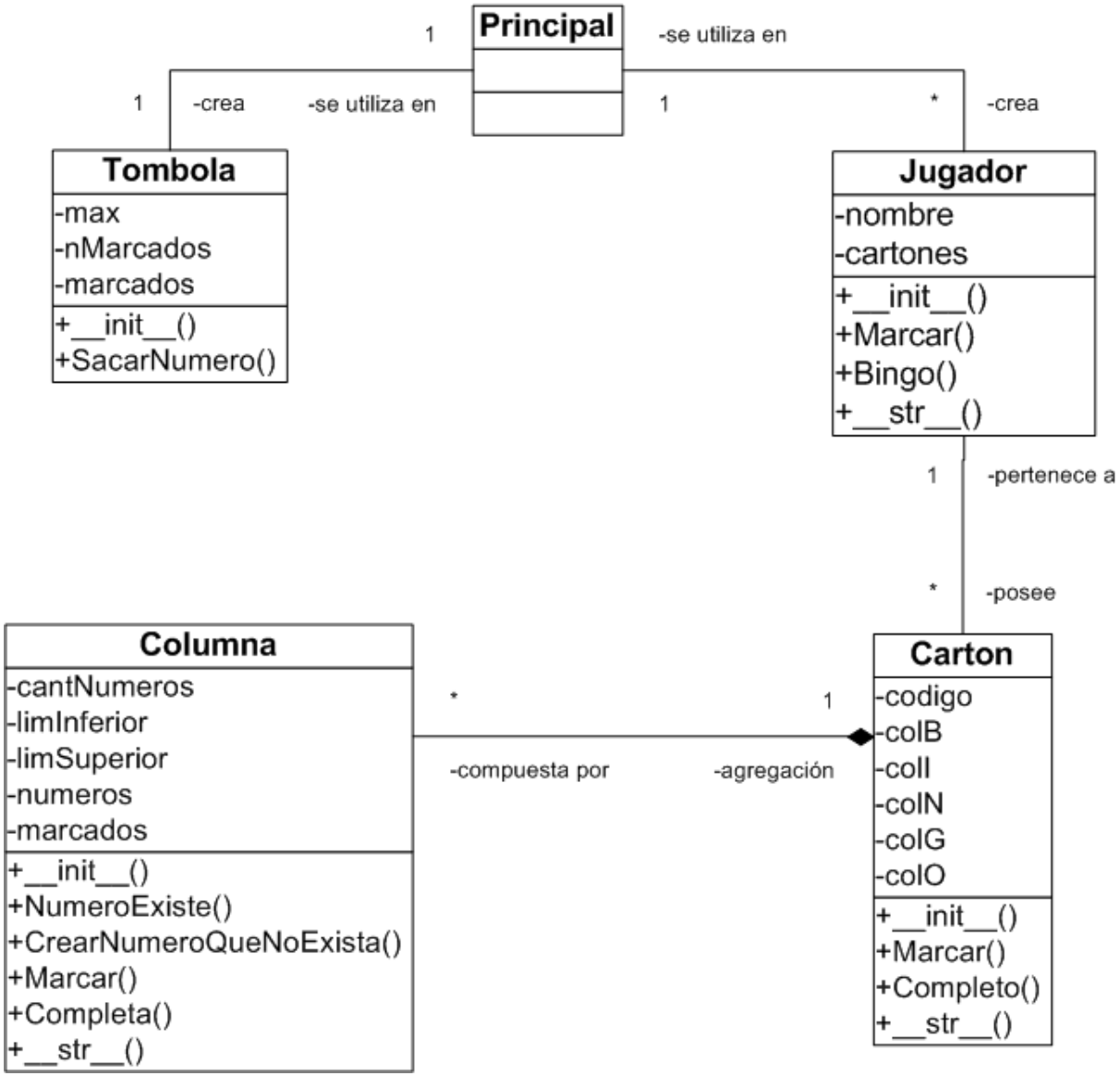
Bingo

- Crear un programa que permita jugar Bingo
  - Cada jugador posee cartones
  - Cada cartón está compuesto de 5 columnas, identificadas por las letras ‘B’, ‘I’, ‘N’, ‘G’ y ‘O’.
  - En cada columna hay 5 números aleatorios (4 en el caso de la columna ‘N’) en un cierto rango.
  - El animador del Bingo extrae números al azar de una tómbola
  - Gana el juego (“hace Bingo”) aquel jugador que completa alguno de sus cartones (hay otras variantes).

Bingo

B	I	N	G	O
9	27	41	59	68
14	26	43	57	70
5	23	FREE	55	66
7	30	36	56	67
6	29	42	58	64

Bingo – clases



Objetivo

- Crear clases `Tombola`, `Columna`, `Carton`, `Jugador`
- Crear una tómbola
- Crear tres jugadores (Hugo, Paco y Luis)
  - Un jugador tiene dos cartones
  - Un cartón tiene 5 columnas (B I N G O)
- Hasta que alguno de los jugadores haga Bingo:
  - Sacar números de la tómbola
  - Mostrar en consola el número que salió
  - Marcar los cartones de los jugadores
- Informar quién ganó el juego

Clase Tómbola

```
class Tombola:
    # Crea una tómbola con números de 1 a max
    def __init__(self, max):
        self.max = max       # número máximo en la tómbola
        self.nMarcados = 0   # cantidad de números entregados
        self.marcados = []   # True si el número i-ésimo ha salido
        ...

    # Retorna un número diferente de la tómbola
    def SacarNumero(self):
        ...
```

Clase Jugador

```
class Jugador:
    # Crea un jugador
    def __init__(self, nombre, nCartones):
        self.nombre = nombre  # nombre del jugador
        self.cartones = []    # lista de cartones
        ...

    # Marca un número (num) en los cartones del jugador
    def Marcar(self, num):
        ...

    # Es True si el jugador tiene alguno de sus cartones lleno
    def Bingo(self):
        ...

    # Retorna un string con los cartones que tiene el jugador
    def __str__(self):
        ...
```

`__str__()`

Sobrecarga de operadores. Método de la clase que permite generar un string. Es invocado por **`print()`**, cuando recibe un objeto de dicha clase como parámetro.

Clase Cartón

```
class Carton:
    # Crea un cartón
    def __init__(self, codigo):
        self.codigo = codigo          # código asignado al cartón
        self.colB = Columna(5, 1,15)
        self.colI = Columna(5,16,30)
        self.colN = Columna(4,31,45)
        self.colG = Columna(5,46,60)
        self.colO = Columna(5,61,75)
        ...

    # Marca un número (num) en el cartón
    def Marcar(self, num):
        ...

    # Es True si el cartón está completo
    def Completo(self):
        ...

    # Retorna un string con los datos almacenados en el cartón
    def __str__(self):
        ...
```

Clase Columna

```
class Columna:
    # Crea una columna de 'n' números entre límites 'inf' y 'sup'
    def __init__(self, n, inf, sup):
        self.cantNumeros = n   # cantidad de números en la columna
        self.limInferior = inf # límite inferior de la columna
        self.limSuperior = sup # límite superior de la columna
        self.numeros = []      # lista: elemento es un número en la columna
        self.marcados = []   # lista: elemento es True si el número está marcado
        ...

    # Es true si el número existe en la columna
    def NumeroExiste(self, num):
        ...
    # Retorna un número válido que no esté en la columna
    def CrearNumeroQueNoExista(self):
        ...
    # Marca un número en la columna
    def Marcar(self, num):
        ...
    # Es True si la columna está completa
    def Completa(self):
        ...
    # Retorna un string con los datos almacenados en la columna
    def __str__(self):
        ...
```

Principal – `principal_bingo.py`

```
from tombola import Tombola
from jugador import Jugador

tomb = Tombola(75)

hugo = Jugador("Hugo", 2)
paco = Jugador("Paco", 2)
luis = Jugador("Luis", 2)

while not hugo.Bingo() and not paco.Bingo() and not luis.Bingo() :
    numero = tomb.SacarNumero()
    print("Número sorteado:", numero)
    hugo.Marcar(numero)
    paco.Marcar(numero)
    luis.Marcar(numero)

print("---------------")
print("Jugador ganador")
if hugo.Bingo(): print(hugo)
if paco.Bingo(): print(paco)
if luis.Bingo(): print(luis)
```