TP 4 - Comisión 3 - David Bedoian (14 - 16)
Introducción a la Programación Orientada a Objetos

Alumna: Valeria Parra
Legajo: 91547/4

A) Fundamento teórico.

1. La **Programación Orientada a Objetos** (POO) es un paradigma de programación, es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos. (https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/)

2. Para declarar una clase (class) en Processing primero debemos escribir class y luego el nombre de nuestra clase, por ejemplo; *class Pelota* con llaves { } y dentro de éstas todo lo que corresponda a esa clase. Seguido a eso obtendremos los objetos *Pelota1* y *Pelota2* los cuales van a responder a la clase Pelota. La clase definirá una serie de atributos o valores propios que cambian a cada objeto, estos valores tienen que estar definidos, junto con los métodos que son un conjunto de instrucciones que cada objeto podrá desarrollar a su manera, el principal de esos objetos es el *Constructor*.

```
class Pelota{
//Atributos de posición
int x, y;
//Atributos de tamaño
int tamaño;
//Constructor
pelota (int a, int b){
tamaño = (int) random (100);
x = a;
y = b;
}
//Método de dibujo
void dibujoPelota (){
ellipse (x, y, tamaño, tamaño);
}
}
```

El setup y el draw van aparte del programa escrito anteriormente.

```
//declaramos un objeto de tipo Pelota
pelota Pelota1;
void setup (){
```

```
size (600, 600);
Pelota1 = new Pelota (300,300);
}
void draw (){
Pelota1.dibujaPelota();
}
```

3. - **Class**: En la programación orientada a objetos, una clase es una construcción que se utiliza como un modelo (o plantilla) para crear objetos de ese tipo. El modelo describe el estado y contiene el comportamiento que todos los objetos creados a partir de esa clase tendrán. (https://dunadigital.com/)

- **Instancia**: Se llama instancia a todo objeto que derive de algún otro. De esta forma, todos los objetos son instancias de algún otro, menos la clase Object que es la madre de todas. (http://www.upv.es/amiga/43.htm)

- **Método**: Son aquellas funciones que permite efectuar el objeto y que nos rinden algún tipo de servicio durante el transcurso del programa. Determinan a su vez como va a responder el objeto cuando recibe un mensaje. (http://www.upv.es/amiga/43.htm)

- **Eventos**: Son aquellas acciones mediante las cuales el objeto reconoce que se está interactuando con él. De esta forma el objeto se activa y responde al evento según lo programado en su código. (http://www.upv.es/amiga/43.htm)

- **Constructor**: Un constructor, en programación orientada a objetos, es un conjunto de instrucciones diseñado especialmente para *inicializar una instancia* de un objeto. Pueden pasar parámetros a un constructor, de la misma forma que una función. (https://lenguajesdeprogramacion.net)

- **Objeto**: Un objeto, en programación orientada a objetos (OOP), es un resumen tipo de datos creado por un desarrollador. Puede incluir múltiples propiedades y métodos e incluso puede contener otros objetos. En la mayoría de lenguajes de programación, los objetos se definen como clases. (https://techlib.net/definition/object.html)

4. La manera en que los objetos pueden comunicarse con otros es mediante el paso de mensajes. Se trata de interfaces específicas que deben ser bien diseñadas a la hora de describir el comportamiento del objeto en la definición de la clase. Son interfaces perfectas porque sirven para lo necesario, ya sea recibir uno o más datos (un número, una letra, o una estructura de datos como un arreglo, etc.) y tal vez generar una respuesta, o simplemente hacer algo internamente por consecuencia de la indicación por parte de otro objeto. Haciendo uso de las interfaces es como los objetos interactúan, esta interacción puede propiciar una colaboración entre los objetos, pudiendo armar un panorama de organización que da lugar a complejas jerarquías que

permiten la delegación de tareas, haciendo más fácil la resolución de problemas grandes.