Programación orientada a objetos

Cristian Rojas
crrojasperez@gmail.com

Paradigmas

- ¿Qué es un paradigma?

Ventajas de los paradigmas

- Entregan una forma (pauta) probada que funciona para resolver ciertos problemas.
- Aplicar un paradigma ampliamente utilizado reduce los riesgos de cometer errores.
- Amplia documentación y ejemplos de uso.
- Múltiples herramientas de apoyo.

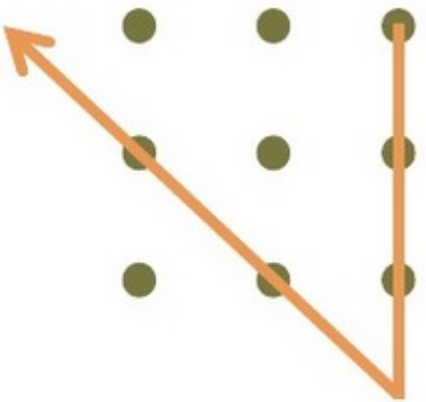
Ejercicio

- Une estos nueve puntos con solo 4 líneas rectas consecutivas, sin levantar el lápiz del papel.
- Si ya conoces la solución espera a que tus compañeros traten de resolverlo por ellos mismos.

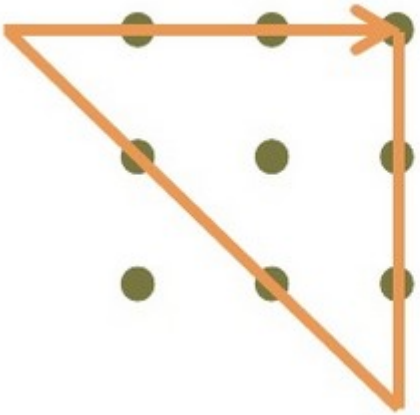
Solución

Trazo 1:

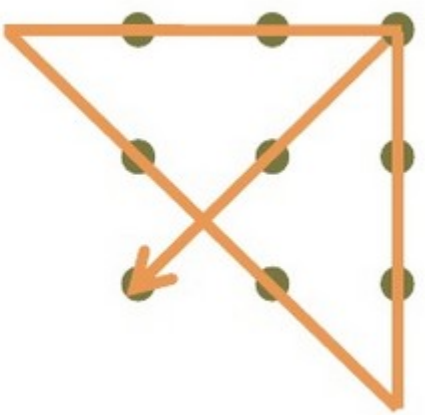
Trazo 2:



Trazo 3:



Trazo 4:



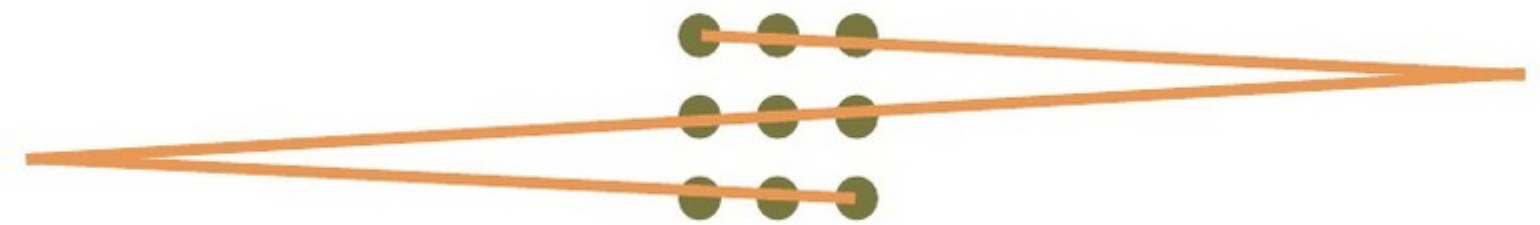
El efecto negativo de los paradigmas

- Cuando nos aferramos de manera inflexible a un paradigma (a una sola forma de ver y actuar), disminuye la capacidad de ver opciones y oportunidades
- Casi siempre hay más de una respuesta correcta.. pero no la vemos por estar cegados con un modelo que nos sirvió, nos fue útil, o nos sigue sirviendo en ciertos casos

Ejercicio

- ¿Cómo unirías los puntos con solo 3 líneas sin levantar el lápiz del papel?

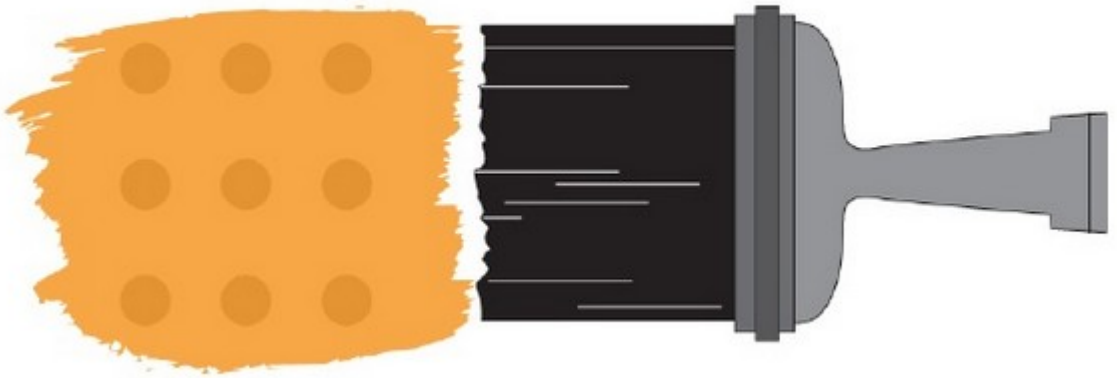
Solución



Ejercicio

- ¿Cómo unirías los puntos con una sola línea?

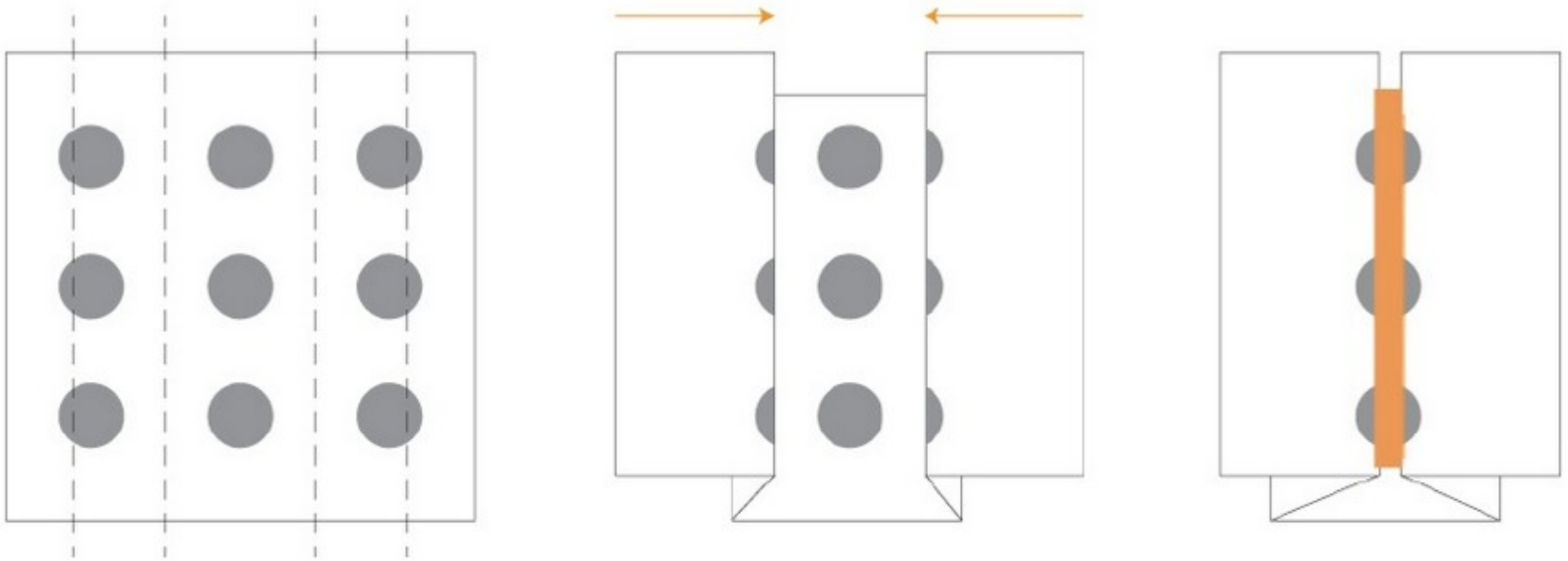
Solución



Ejercicio

- ¿Propuesta alternativa para unir los 9 puntos con una sola línea recta?

Solución



Puedes doblar la hoja en la que estén los 9 puntos,
de manera tal que las 3 hileras de puntos queden "sobrepuestas".
Después los unes trazando una sola línea.

Ejercicio

- Una persona quiere entrar a una orden secreta llamada los “mágios”. Para esto espía en la entrada a los integrantes que entran. Para ingresar solo deben responder una pregunta que les hace un guardia.
  - Llega el primer integrante, el guardia le dice “24”. Él responde “12” y lo dejan pasar.
  - Llega el segundo integrante, el guardia le dice “18”. Él responde “9” y lo dejan pasar.
  - Llega el tercer integrante, el guardia le dice “14”. Él responde “7” y lo dejan pasar.
  - Llega el cuarto integrante, el guardia le dice “8”. Él responde “4” y lo dejan pasar.
- Convencido de que ya ha resuelto el acertijo se atreve a avanzar hacia la puerta.
  - El guardia le dice “cuatro”. El responde “dos”. El guardia grita “Traidor” y lo sale persiguiendo.
- ¿Cuál era la respuesta correcta?.

Paradigmas de programación

- Es una propuesta tecnológica adoptada por una comunidad de programadores cuyo núcleo central es incuestionable en cuanto a que únicamente trata de resolver uno o varios problemas claramente delimitados.
- Tipos de paradigmas de programación
  - Imperativo o por procedimientos. (C, Basic, Pascal)
  - Funcional. (Lisp, scheme, haskell, python)
  - Lógico. (Prolog)
  - Orientado a objetos. (Jaca, C++, Python, etc.)
  - Programación dinámica.
  - Etc.
- En la práctica es habitual mezclar paradigmas (Multiparadigma)

Paradigma de programación Orientado a Objetos (POO)

- Conceptos relevantes del mundo real (para nuestro problema) se modelan a través de clases y objetos.
- El programa consiste en una serie de interacciones entre estos objetos
- Conceptos fundamentales
  - Clase
    - Definición de las propiedades y comportamiento de un objeto conreto. Como una plantilla. (Alumno, casa, animal)
  - Herencia
    - Las clases pueden heredar los atributos y propiedades de otra. (Alumno hereda las propiedades de persona)
  - Objeto
    - Instancia concreta de una clase. (Alumno cristian rojas con rut 14.083.018)
  - Método
    - Definen lo que el objeto puede hacer (acciones).

Propiedades de la POO

- Abstracción
  - Se refiere al énfasis en el **¿Qué hace?** Más que en el **¿Cómo lo hace? (Caja negra)**
- Encapsulamiento
  - Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción
  - **Principio de ocultación**: Los objetos están aislados del exterior. Estos exponen interfaces para que otros objetos puedan interactuar con ellos.
- Modularidad
  - Subdividir una aplicación en partes más pequeñas, independientes y conectadas.
- Polimorfismo
  - Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre. (“hola” + “mundo”, 2 + 3, 5.4 + 3.9)
- Herencia
  - Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenece.

Programación Orientada a **OBJETOS**!!

- Las clases son solo plantillas.
- La POO se trata de la interacción entre **objetos**.
- Los **objetos** son entidades (**instancias** de una clase) que tienen un determinado estado, comportamiento (método) e identidad:
  - El estado está compuesto de datos o informaciones; serán uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
  - El comportamiento está definido por los métodos o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.
  - La identidad es una propiedad de un objeto que lo diferencia del resto; dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

POO en Python

- Para definir un método o propiedad de un objeto como privado se deben anteceder dos guiones bajos (underscore) a su nombre
  - __variable = 5
  - __metodo():
- Para hacer referencia a un método o propiedad de una clase dentro de la misma clase es necesario utilizar el parámetro self.
  - self.__metodo()
  - self.__variable = 5
- El encapsulamiento en python se puede simplificar mediante propiedades, que abstraen al usuario del hecho de que está utilizando métodos tras bambalinas.

```
class Fecha(object):
    def __init__(self):
        self.__dia = 1

    def getDia(self):
        return self.__dia

    def setDia(self, dia):
        if dia > 0 and dia < 31:
            self.__dia = dia
        else:
            print "Error"

    dia = property(getDia, setDia)

mi_fecha = Fecha()
mi_fecha.dia = 33
```

Objetos

¿Cuáles son sus propiedades?
¿Cuáles son sus funciones?

Nombre: Auto

Propiedades:

- Color: Plateado
- Marca: Nissan
- Tipo: Sedan
- Llantas: si
- Tipo combustible: Bencina
- Litros de combustible: 10
- Rendimiento: 12 km/lt
- etc

Funciones (métodos):

- Encender
- Mover
- frenar
- Cargar bencina

POO en Python

- Para definir una clase, basta con poner su nombre tras la palabra reservada **class** y lo que sigue forma parte de la definición de la clase.
- Generalmente, la definición de la clase consiste en una serie de funciones que serán los métodos de la clase.
- Todos los métodos tienen **self** como primer parámetro. Este es el parámetro con el que se pasa el objeto. El nombre **self** es convencional, puede ser cualquier otro.
- Existe un método especial __**init**__. A este método se le llama cuando se crea un objeto de la clase.
- El método __**init**__ puede tener parámetros adicionales.

POO en Python

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

```
class Mapping:
    def __init__(self, iterable):
        self.items_list = []
        self.__update(iterable)

    def update(self, iterable):
        for item in iterable:
            self.items_list.append(item)

    __update = update   # private copy of original update() method

class MappingSubclass(Mapping):

    def update(self, keys, values):
        # provides new signature for update()
        # but does not break __init__()
        for item in zip(keys, values):
            self.items_list.append(item)
```

```
class Bag:
    def __init__(self):
        self.data = []
    def add(self, x):
        self.data.append(x)
    def addtwice(self, x):
        self.add(x)
        self.add(x)
```

```
class Ejemplo:
    def publico(self):
        print "Publico"

    def __privado(self):
        print "Privado"

ej = Ejemplo()
ej.publico()
ej.__privado()
```

```
class Fecha():
    def __init__(self):
        self.__dia = 1

    def getDia(self):
        return self.__dia

    def setDia(self, dia):
        if dia > 0 and dia < 31:
            self.__dia = dia
        else:
            print "Error"

mi_fecha = Fecha()
mi_fecha.setDia(33)
```

Ejercicio 1

- Cree una clase auto con sus atributos y métodos como lo vimos en la diapositiva anterior
  - Atributos:
    - Kilometraje
    - Bencina
    - Rendimiento
    - Encendido: (True, False)
  - Métodos:
    - Encender
    - Apagar
    - Mover
    - Obtener kilometraje
    - Obtener bencina
    - Cargar bencina
  - Debe considerar:
    - Que haya bencina para mover el auto la cantidad de Km solicitada
    - Al cargar bencina el motor debe estar apagado

Herencia

- En un lenguaje orientado a objetos cuando hacemos que una clase (subclase) herede de otra clase (superclase) estamos haciendo que la subclase contenga todos los atributos y métodos que tenía la superclase. No obstante al acto de heredar de una clase también se le llama a menudo “extender una clase”.

```
class DerivedClassName(BaseClassName):
    <statement-1>
    .
    .
    .
    <statement-N>
```

Herencia múltiple

- En Python, a diferencia de otros lenguajes como Java o C#, se permite la herencia múltiple, es decir, una clase puede heredar de varias clases a la vez.

```
class Terrestre:
    def desplazar(self):
        print "El animal anda"

class Acuatico:
    def desplazar(self):
        print "El animal nada"

class Cocodrilo(Terrestre, Acuatico):
    pass

c = Cocodrilo()
c.desplazar()
```

- En el caso de que alguna de las clases padre tuvieran métodos con el mismo nombre y número de parámetros las clases sobreescribirían la implementación de los métodos de las clases más a su derecha en la definición.

Ejercicio 2

- Cree una clase Camión que herede de Vehículo y además tenga los atributos:
  - Peso: (float)
  - Ruedas: (int)
  - Acoplados: (lista)
- Acplado es un objeto que tiene la propiedad ruedas (int), peso (float) y carga (text)
- Debe implementar los métodos:
  - agregar_acoplado
  - quitar_acoplado
  - obtener_acoplados
  - obtener_ruedas
  - obtener_peso

Ejercicio 3

- Crear una clase persona con los siguientes atributos y métodos.
  - Atributos: rut, nombre, fecha nacimiento, género
  - Métodos: obtener edad
- Crear una clase nota con los siguientes componentes
  - Atributos: valor, ponderación, ramo, carrera
  - Métodos: obtener_valor, obtener_ponderacion, modificar
- Crear una clase alumno que herede de Persona con los siguientes componentes
  - Atributos: correo, carrera, promoción, notas (lista)
  - Métodos: agegar nota, PGA, promedio por ramo,
- Programe ejemplos de utilización de estas clases