Universidad Nacional de Quilmes

</CPI>
01010101.01001110.01010001

Programación con Objetos 1

"Colecciones"

Definiciones, Usos, Mensajes y Ejemplos

ACERCA DE UNA COLECCIÓN...

¿Que es una colección?

1. Una Colección es un **Objeto** que permite agrupar a un conjunto de **Objetos**. Contamos con distintas clases para representar distintos tipos de colecciones y trabajamos con ellas creando instancias de las mismas.

    unaColeccion <- Colección new.

    Ej: *unaColecciónOrdenada <- OrderedCollection new.*

¿Para que sirve?

2. Las colecciones nos resultan de utilidad porque nos permiten agrupar objetos, para luego poder operar sobre un elemento en particular, sobre algunos elementos seleccionados mediante un filtro, o sobre una colección como conjunto.

¿Qué objetos podemos agrupar?

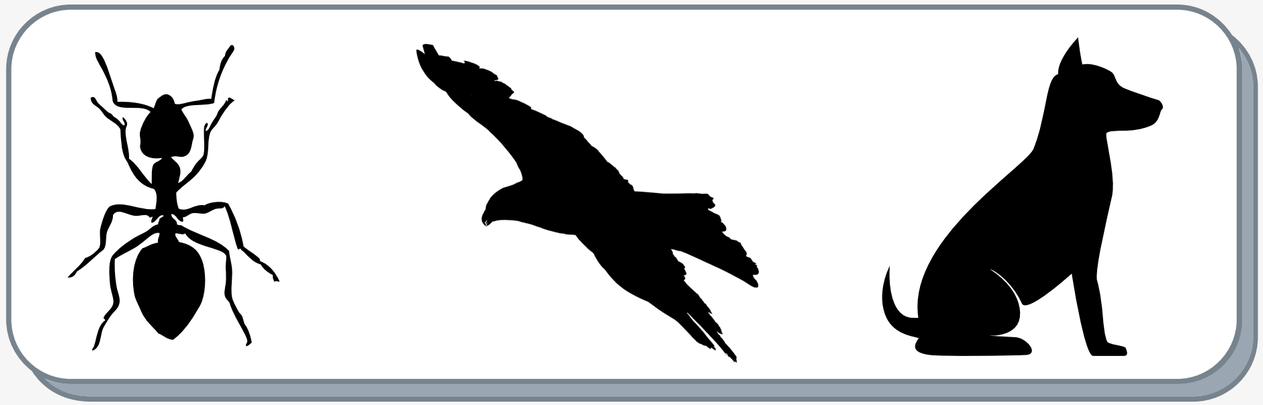
3. En Smalltalk no hay limitaciones con respecto a los objetos que podemos agregar en una misma colección.

Importante: Siempre vamos a manipular instancias de cada clase de Colección

¿QUÉ OBJETOS SE PUEDEN AGRUPAR?

En realidad... Se puede añadir cualquier **Objeto**, preferentemente que tengan el mismo comportamiento (es decir, que cada objeto de la colección compartan el mismo protocolo).

collection:



Todos saben responder al mensaje #esUnAnimal.
Es decir, tienen al menos un comportamiento igual.

Comportamientos de los objetos en la coleccion

¿Cómo operamos sobre las colecciones?

Como las colecciones también son **Objetos**, por supuesto que pueden responder distintos **mensajes** para **obtener información**, como así también para **operar** sobre ellas. Algunos de ellos son:

- Agregar Elementos
- Quitar Elementos
  (El elemento debe estar en la colección para quitarlo)
- Conocer si contiene un elemento
- Conocer si está vacía
- Conocer la cantidad de elementos

```
unaColeccionInicial add: unObjeto.
unaColeccionInicial addAll: unaColección.

unaColeccionInicial remove: unObjeto.
unaColeccionInicial removeAll: unaColección.

unaColeccionInicial includes: unObjeto.

unaColeccionInicial isEmpty.
unaColeccionInicial  notEmpty.

unaColeccionInicial size.
```

Mensajes que podemos enviar a las colecciones

Otros mensajes útiles...

- #sum: / Sumar un valor de los elementos de la colección.
  Por ejemplo: Números u otros objetos de la colección que comprendan el mensaje + , es decir que pueden sumarse entre si.

- #detectMin: / Devolver el elemento mínimo de una colección.
  (Por ejemplo La Golondrina con menor edad)

- #detectMax: / Devolver el elemento máximo de una colección.
  (Por ejemplo La Golondrina con mayor edad)

- #detect: ifNone: / Devolver el elemento que cumple con la condición u otro en el caso de que no se encuentre el elemento buscado.

- #sum: ifEmpty: / Sumar elementos de una colección y si ésta está vacía realizar otra acción.

Mensajes que podemos enviar a las colecciones

Otros mensajes útiles...

- colección detect: [: elemento | elemento mensaje]
- #select: / Seleccionar los elementos que cumplen con un criterio y devolver una colección con ellos.
- #reject: / Quitar los elementos que cumplen con un criterio y devolver una colección sin ellos.
- #collect: / Recolectar el resultado de hacer algo con cada elemento
- #do: / Hacer algo con cada objeto de la colección
- #average / Calcular el promedio

Mensajes que podemos enviar a las colecciones

TIPOS DE COLECCIONES

Instanciar cada una de ellas.

Al momento de instanciar una colección tenemos que decidirnos por una colección concreta.

Bag

```
unaColeccionTipoBag ← Bag new.
```

Representa a una bolsa, como las bolsas de compras. Pueden tener elementos repetidos, pero no tienen un orden establecido en la colección.

Set

```
unaColeccionTipoSet ← Set new.
```

Representa un conjunto. Por definición de lo que es un conjunto, cada elemento en él es único, es decir, no pueden tener elementos repetidos; los sets no tienen un orden establecido.

Instanciar una colección según su tipo

Ordered Collection

```
unaColeccionOrdCollection ← OrderedCollection new.

unaColeccionOrdCollection add:objeto1 .
unaColeccionOrdCollection add:objeto2 .
unaColeccionOrdCollection first  .  "Retorna el primer elemento:  objeto1"
```

Representa una colección ordenada dependiendo de cómo agregamos los objetos a la colección. Este tipo de Colección tiene orden y permite objetos repetidos.

Sorted Collection

```
sortedCollection ← SortedCollection new.
ordenadosPorPrecio ← SortedCollection sortBlock: [:unProducto | unProducto esMasCaroQue: otroProducto].
```

Representa una colección ordenada dependiendo del criterio que dispongamos.
Si no definimos el criterio, SortedCollection ordena los elementos por su “orden natural” (significa que los ordenará de menor a mayor). Este tipo de Colección tiene orden y permite objetos repetidos.

Instanciar una colección según su tipo

Dictionary

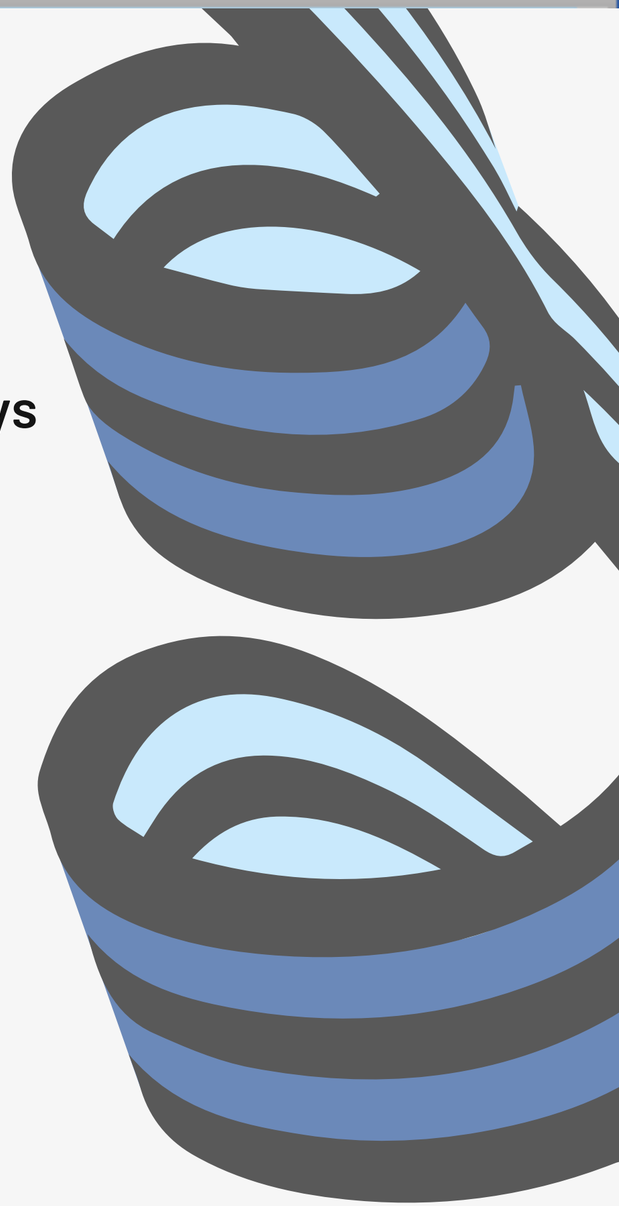
Representa una colección de objetos que no tiene "keys" repetidos ni orden.
Estos objetos están asociados a un key que nos permitirá acceder al objeto.
Dato Importante: Lo que no se repiten son las keys, pero asociadas a distintas keys puede haber un mismo objeto varias veces.

```
unaColeccionDiccionario ← Dictionary new.
unaColeccionDiccionario at: #key1 put: unObjeto.
```

Array

Representa una colección de objetos que posee un tamaño definido en su inicialización; para añadir un elemento tengo que indicar la posición en la cual quiero que se guarde.

```
coleccionTipoArray ← Array  ofSize: 8.      "elijo que el tamaño del array sea 8"
coleccionTipoArray at: 2 put: objetoAñadir.  "indico la posicion ( 2 ) en donde voy a guardar mi objeto"
```



Instanciar una colección según su tipo

EJEMPLO PÁJAROS - OPERACIONES CON COLECCIONES

Suponemos que existe una colección llamada pájaros con objetos Pájaro.
No existen pájaros exactamente iguales ni tampoco podemos tenerlos con un orden específico.

¿Qué colección podría ser la indicada?

...

Ejemplo para entenderlo mejor

EJEMPLO PÁJAROS - OPERACIONES CON COLECCIONES

Suponemos que existe una colección llamada pájaros con objetos Pájaro.
No existen pájaros exactamente iguales ni tampoco podemos tenerlos con un orden específico.

¿Qué colección podría ser la indicada?

```
pajaros ← Set new.
```

Ejemplo para entenderlo mejor

Operaciones sobre los elementos de una colección

pajaros do: [:unPajaro | unPajaro volar: 100kilometer].

“Le enviamos el mensaje #volar: a cada pájaro de la colección pájaros ”

pajaros anySatisfy: [:unPajaro | unPajaro estaDebil].
pajaros allSatisfy: [:unPajaro | unPajaro estaDebil].

“Indica si al menos uno o todos los elementos cumple con la condición”

pajaros detect: [:unPajaro | unPajaro estaDebil].

“Retorna el primer objeto de la colección que cumpla con la condición.”
Notar que arroja una excepción cuando no encuentra un objeto que cumpla con la condición.

Operaciones sobre los elementos de una colección

pájaros anyOne.

“Retorna cualquier objeto de la colección.”

pajaros select: [:unPajaro | unPajaro estaDebil].

“Retorna una colección con los objetos que cumplan con la condición “

pájaros sum: [: unPajaro | unPajaro cantidadPichones].

“Retorna la suma de la cantidad total de pichones de todos los pájaros.”

Notar que si la colección está vacía falla, para eso podemos usar #sum:ifEmpty: ”

PEQUEÑO RESUMEN

Tipos de Colecciones

	Bag	Set	OrderedCollection	SortedCollection	Dictionary
Repetidos	Si	No	Si	Si	No(Keys)
Orden	No	No	Si	Si	No

- **Bag** es la representación de una bolsa.
- **Set** es la representación de un conjunto.
- **OrderedCollection** ordena los objetos por orden de adición.
- **SortedCollection** ordena los objetos basado en un criterio.
- **Dictionary** guarda objetos con una clave asociada.
- **Array** representa una colección de objetos que posee un tamaño definido.
  (Importa el orden y puede tener elementos repetidos)

¡ES TODO POR HOY!

Universidad Nacional de Quilmes

</CPI>
01010101.01001110.01010001

¡GRACIAS!

CONTESTAMOS PREGUNTAS, ACEPTAMOS SUGERENCIAS O COMENTARIOS