Programación Orientada a Objetos

Clase 5 – Colecciones

UTN - La Plata

Colecciones

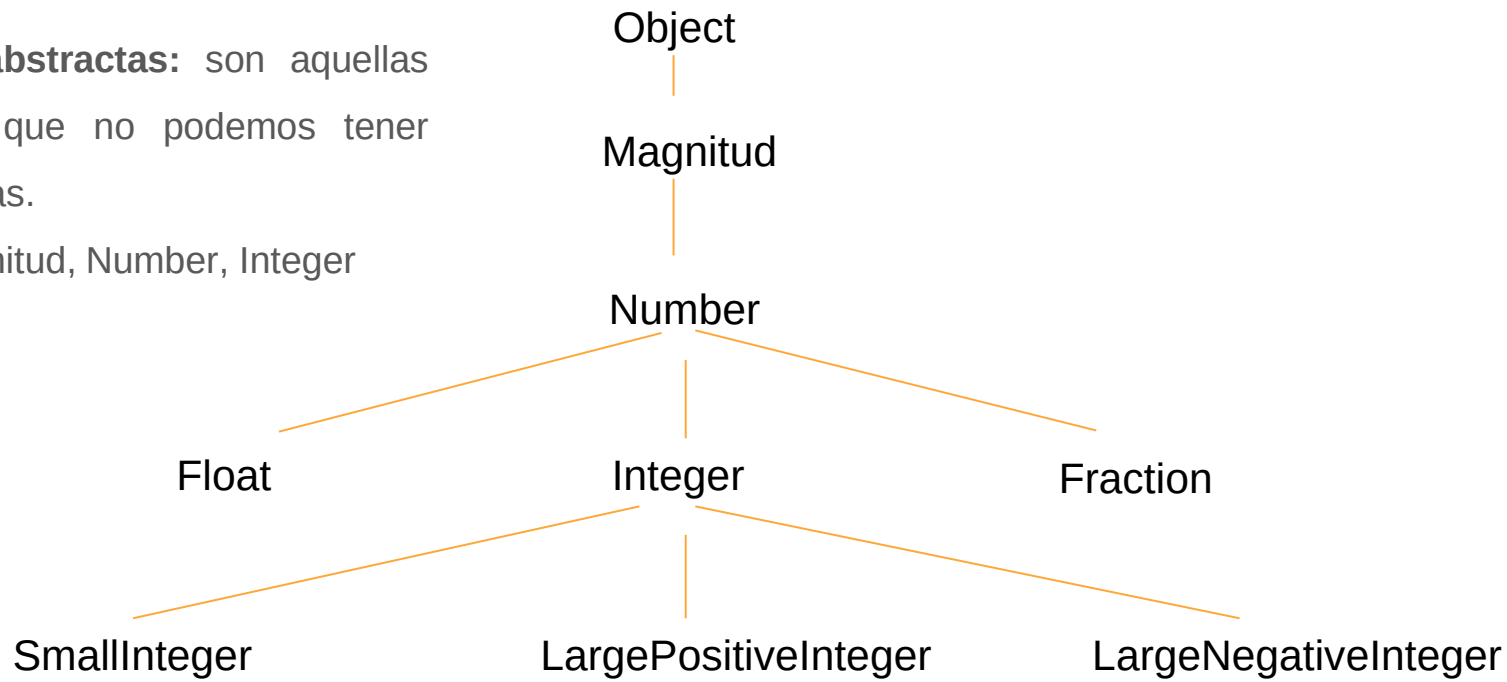
1) Jerarquías Magnitud y Collection en Smalltalk
2) Tipos de colecciones:
    a) Array
    b) OrderedCollection
    c) SortedCollection
    d) Dictionary
3) Testeo y conversión de colecciones
4) Enumeradores de colección
5) Actividad 5

1) Jerarquías en Smalltalk

Jerarquía Magnitud

Clase abstractas: son aquellas de las que no podemos tener instancias.

Ej: Magnitud, Number, Integer



1) Jerarquías en Smalltalk

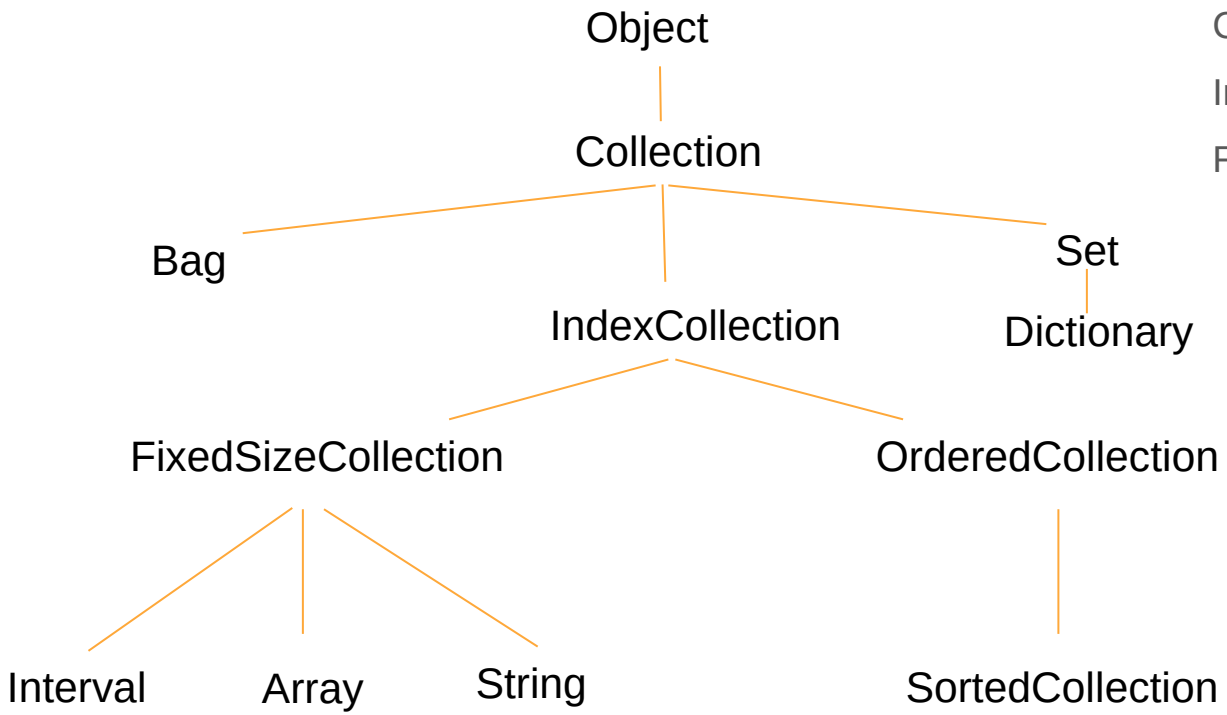
Jerarquía Collection

Colecciones abstractas:

Collection,

IndexedColecction,

FixedSizeCollection



2) Tipos de Colecciones

No nos interesa cómo están implementadas las colecciones sino lo que podemos hacer con ellas.

Bag: son colecciones no ordenadas que puede contener elementos repetidos.

Set: son colecciones no ordenadas que no permiten elementos repetidos.

La clase **Interval** representa rangos de números.

Por ejemplo, el intervalo de números de 1 a 100 se define de la siguiente manera:

Interval from: 1 to: 100.

Algunas colecciones en Smalltalk pueden ser heterogéneas. Por ejemplo: Array, OrderedCollection, SortedCollection.

String es homogéneo, está formado sólo por caracteres.

2) Tipos de Colecciones

a) Array (subclase de FixedSizeCollection)

```
|a|
a:=Array new:3.
a at:1 put:5.
a at:2 put:8.
a at:3 put:2.
a inspect.
(a at:2) inspect.
```

Muestra #(5 8 2)

Muestra 8

El array no admite el add: y tampoco el remove:

2) Tipos de Colecciones

b) OrderedCollection (subclase de IndexedCollection)

```
|oc|
oc:=OrderedCollection new.
oc add:5.
oc add:8.
oc add:2.
oc inspect.                          Muestra   #(5 8 2)
(oc at:2) inspect.                   Muestra   8
(oc at:1 put: 23)  inspect.          Muestra   #(23 8 2)
```

Para insertar	**Para recuperar**	**Para eliminar**
add:unObjeto	first	removeFirst
addFirst:	last	removeLast
addLast:	at:	remove:
add: before:		remove: ifAbsent:
add: after:		
at: put:		

La OrderedCollection admite el **at:** y el **at: put:**, con ellos podemos manejar la colección en forma indexada. Solo pueden usarse si la posición ya tiene asignado contenido.

2) Tipos de Colecciones

c) SortedCollection (subclase de OrderedCollection)

|sc|

sc:=SortedCollection new. /*crea una colección vacía cuyos elementos por defecto serán ordenados de menor a mayor cuando se agreguen, solo en caso de ser objetos de clases simples como números y strings*/

sc add:5. El add agrega un elemento a la SortedCollection en forma ordenada.

sc add:8.

sc add:2.

sc inspect. Muestra #(2 5 8)

(sc at:2) inspect. Muestra 5

2) Tipos de Colecciones

Si se necesita que los elementos estén en orden descendente hay que usar el método de clase sortBlock:[ ] para crear la colección con un criterio de orden distinto

```
s:=SortedCollection sortBlock [:x :y| x >=y].
sc add:5.
sc add:8.
sc add:2.
sc inspect.
```

Muestra #(8 5 2)

2) Tipos de Colecciones

En caso de que sean objetos compuestos, como libros, personas, productos, etc. debe usarse el método de clase sortBlock:[ ] e indicar el criterio de ordenamiento:

s:=SortedCollection sortBlock [:x :y| x verNombre <= y verNombre].

En la clase SortedCollection también existe el método de instancia sortBlock: [ ] que permite reordenar una colección ya cargada bajo otro criterio.

Por ejemplo, podemos cambiar el ordenamiento de s:

s:= s sortBlock:[:x :y| x verPrecio >= y verPrecio].

Este método cambia el criterio de ordenamiento y reordena todos los elementos de s.

2) Tipos de Colecciones

d) Dictionary (subclase de Set)

```
|d|
d:=Dictionary new at:'lunes' put:'Física';
                 at:'martes' put:'Computación';
                 at:'miércoles' put:'Matemática';
                 at:'jueves' put:'Computación';
                 at:'viernes' put:'Matemática';
                 yourself.
d inspect.
```

Muestra la asociación entre clave y valor

Puedo también agregar una Asociación creada previamente. Una Asociación está formada por una clave y un valor. La forma de agregar una asociación es: d add: unaAsociación, previamente habiendo creado dicha asociación.

3) Testeos y Conversión de colecciones

Testeos de colecciones:

size

isEmpty

includes:unObjeto

ocurrencesOf: un Objeto

Conversión de colecciones:

asSet

asBag

asArray

asOrderedCollection

asSortedCollection

Ejemplo de conversión de colecciones:

```
|oc|
oc:=OrderedCollection new.
oc add:5.
oc add:8.
oc add:2.
oc inspect.
```

sc:= oc asSortedCollection. "solo sirve si la colección contiene nros o strings"

sc inspect. Muestra #(2 5 8)

otra:=col asSortedCollection [:a :b | a verAutor > b verAutor].

"convierte a SortedCollection y fija el criterio de ordenamiento, cuando la colección contiene objetos no simples"

4) Enumeradores de colecciones

Enumeradores de colecciones:

do: itera sobre cada elemento de la colección.

Ejemplo: col **do**: [:pieza| pieza reset].

es un objeto

No se debe agregar ni remover un elemento dentro de un do:

Porque trabaja internamente con el tamaño de la colección.

to: do: repite un número conocido de veces un proceso.

Ejemplo: |suma col|

suma:=0.

col:=#(1 3 6 8).

1 **to**:(col size) **do**:[:i |suma:=suma + col at:i].

es un índice

4) Enumeradores de colecciones

Uso del select, reject, collect y detect

Todos ellos están implementados internamente con un do:

select col1:= alumnos select:[:alu| alu verNota=8].

Retorna en otra colección los alumnos con nota igual a 8

reject col2:= alumnos reject:[:alu| alu verNota=8].

Retorna en otra colección los alumnos con nota distinto a 8

collect col3:= alumnos collect:[:alu| alu verNombre].

Retorna en otra colección los nombres de los alumnos

detect a:= alumnos detect:[:alu| alu verNota=8] ifNone:[nil].

Retorna el primer alumno con nota igual a 8 o nil si no hay ninguno

a modNota:10. “modifica la nota de ese alumno encontrado”

5) Actividad 5

Dado el siguiente diccionario:

```
|d|
d:=Dictionary new at:’gorrión’ put:’pájaro’;
at:’olmo’ put:’árbol’;
at:’golondrina’ put:’pájaro’;
at:’sauce’ put:’árbol’;
yourself.
```

Investigar que retornan o qué modifican las siguientes líneas de código

```
rta1:=d at:’golondrina’.
rta2:=d at:’gorrión’.
rta3:=d at:’pavo’.
d at:’fuego’ ifAbsent:[‘no lo se’].
d removeKey:’fuego’ ifAbsent:[^nil].
```