

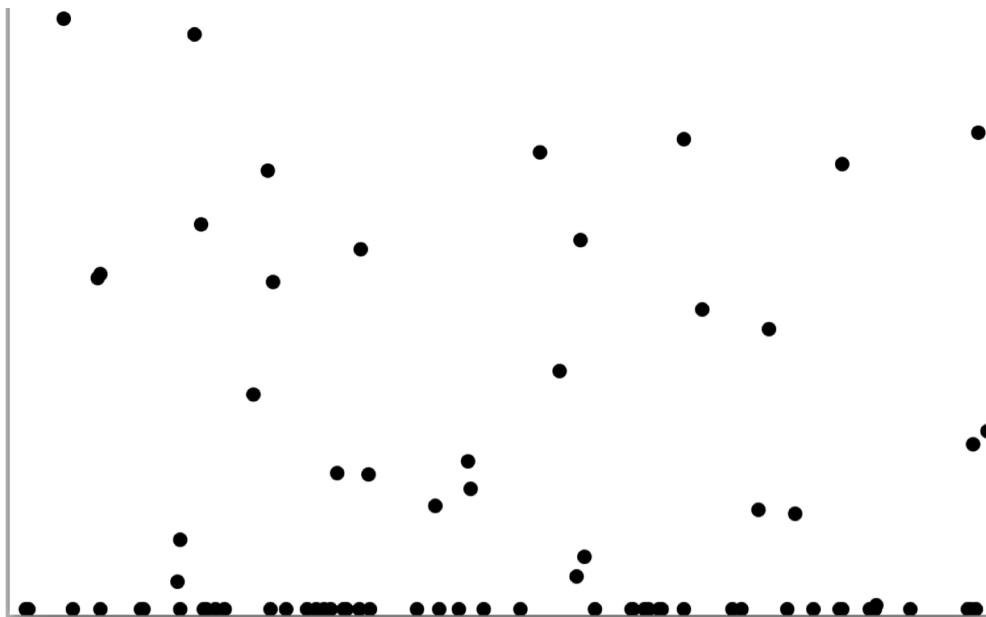


ArrayLists

CS Bridge

Ana Saavedra
Based on slides by Chris Piech (¡Mil gracias!)

¿Por qué este programa es difícil de escribir?



El poder de las colecciones en el mundo



<http://tuplanetavital.org/wp-content/uploads/2013/02/caravana-hormigas-arrieras2.jpg>



<https://headbng.com/wp-content/uploads/2017/07/Abejas.jpg>

El poder de las colecciones en el mundo



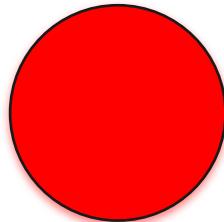
<http://www.filatelialopez.com/images/colkookaburra1.gif>



<https://www.soane.org/sites/default/files/banners/soane-research-library-books.jpg>

Y nosotros... ¿qué hemos hecho hasta hora?

Trabajar con datos individuales....



¿Y entonces?

- ★ ¿Qué pasa cuando tenemos un número infinito de variables?
- ★ El poder de la programación está en la habilidad de trabajar con colecciones de datos.

**Vamos a aprender sobre una
clase especial que nos permite
trabajar con colecciones.**

ArrayLists

¿Qué es?

- Es un **objeto especial** que es utilizado para guardar una lista de otros objetos.
- Te deja poner una colección de objetos en un solo “lugar”.
- Es una lista **ordenada** y **redimensionable** de información (e.g. Strings, int, double, GObject).

Características

- Son **homogéneas**: guardan el **mismo tipo** de variables.
 - Para usarlas deben: Import `java_util*`

Se puede **agregar** y **remover** elementos (además de otras funcionalidades súper chéveres :-)).

- Por ejemplo: agregar elementos al final de lista.
- Las operaciones son implementadas como métodos.

¿Qué tan grande es una lista?



Sintaxis - Nuestra primera ArrayList

```
ArrayList<String> miArrayList = new ArrayList<String>();
```

Encabezado de los métodos

```
ArrayList<String> miArrayList = new ArrayList<String>();
```



Tipo de objeto que mi
ArrayList va a guardar

Encabezado de los métodos

```
ArrayList<String> miArrayList = new ArrayList<String>();
```



Nombre

Encabezado de los métodos

```
ArrayList<String> miArrayList = new ArrayList<String>();
```



El mismo tipo acá pero
debe tener un ().

¿Qué podemos hacer?

```
ArrayList<String> miArrayList = new ArrayList<String>();  
  
// Adiciona elementos al final  
miArrayList.add("feliz");
```

¿Qué podemos hacer? - Adicionar

```
ArrayList<String> miArrayList = new ArrayList<String>();  
  
// Adiciona elementos al final  
miArrayList.add("feliz");  
miArrayList.add("dia");  
miArrayList.add("de");  
miArrayList.add("muchas");  
miArrayList.add("inspiracion");
```

¿Qué podemos hacer? - Acceder

```
ArrayList<String> miArrayList = new ArrayList<String>();  
  
// Adiciona elementos al final  
imprimir(miArrayList.get(0)); //Imprime "feliz"  
imprimir(miArrayList.get(1));//Imprime "dia"  
imprimir(miArrayList.get(2));//Imprime "de"  
imprimir(miArrayList.get(3));//Imprime "muchas"  
imprimir(miArrayList.get(4));//Imprime "inspiracion"
```

¿Qué podemos hacer? - OJO

```
// Tipo equivocado - Estamos en problemas - no  
va a compilar  
SEtiqueta etiqueta = new SEtiqueta("feliz dia");  
miArrayList.add(etiquetalabel);
```

¿Qué podemos hacer? - OJO

```
// Index inválido- Estamos en problemas -  
Crashes! - IndexOutOfBoundsException  
  
imprimir(miArrayList.get(5)); // Oppss está vacío...
```

Qué podemos hacer? - Ciclos

```
// Acceder a los elementos por su index  
(empezando en 0)  
for (int i = 0; i < miArrayList.size(); i++) {  
    String str = miArrayList.get(i);  
    println(str);  
}  
// feliz  
// dia  
// de  
// mucha  
// inspiracion
```

¿Qué podemos hacer? - Remove

- Un elemento a la vez
- Remover: mueve a TODOS los elementos
 - No les gustan los espacios en blanco.

```
miArrayList.remove(2);
```

Métodos de la Clase ArrayList

boolean add(<T> element)

Adiciona un nuevo elemento al final de `ArrayList`; el valor que retorna siempre es `true`.

void add(int index, <T> element)

Inserta un nuevo elemento en la `ArrayList` antes de la posición especificada en el `index`.

<T> remove(int index)

Remueve el elemento en la posición especificada y retorna el valor.

boolean remove(<T> element)

Remueve la primera instancia del `element`, si aparece; devuelve `true` si se encuentra un “match”.

void clear()

Remueve todos los elementos de la `ArrayList`.

int size()

Retorna el número de elementos en la `ArrayList`.

<T> get(int index)

Retorna el objeto en el `index` especificado.

<T> set(int index, <T> value)

Pone el nuevo valor en el `index` especificado y devuelve el valor anterior.

int indexOf(<T> value)

Retorna el `index` de la primera ocurrencia del valor especificado, o `-1` si no se encuentra.

boolean contains(<T> value)

Retorna `true` si la `ArrayList` contiene el valor especificado.

boolean isEmpty()

Retorna `true` si la `ArrayList` no tiene ningún elemento.



Primitivos

Las ArrayLists y los primitivos tienen una historia de amor que requiere de más cuidado.

Cuál es ese cuidado?

Si quieres usar una variable de tipo primitivo, lo tienes que indicar en el encabezado.

- La conversión sucede de forma automática.



Primitivos

Primitivo	Clase “Wrapper”
int	Integer
double	Double
boolean	Boolean
char	Character



ArrayList con Wrappers

```
// Utilicen la clase Wrapper cuando estén  
haciendo un ArrayList  
ArrayList<Integer> numList = new ArrayList<Integer>();  
  
numList.add(123);  
numList.add(546);  
  
int primerNum = numList.get(0);          // 123  
int segundoNum = numList.get(1); // 456
```

La conversión sucede de forma
automática

Estructura de Datos

Operación

Hacer una nueva

```
ArrayList<String> list = new ArrayList<String>();
```

¿Qué tan larga es?

```
list.size();
```

¿Necesitan un elemento?

```
list.get(i);
```

¿Adicionar un elemento?

```
list.set(i, value);  
list.add(value);
```

¿Ciclo?

```
for (int i = 0; i < miArrayList.size(); i++)
```

**iHagan lo que les
apasiona!**

Caminen con los
ojos ABIERTOS

PERSISTENCIA \wedge 2

**Para que se sigan inspirando
en esta aventura de la
programación....**
