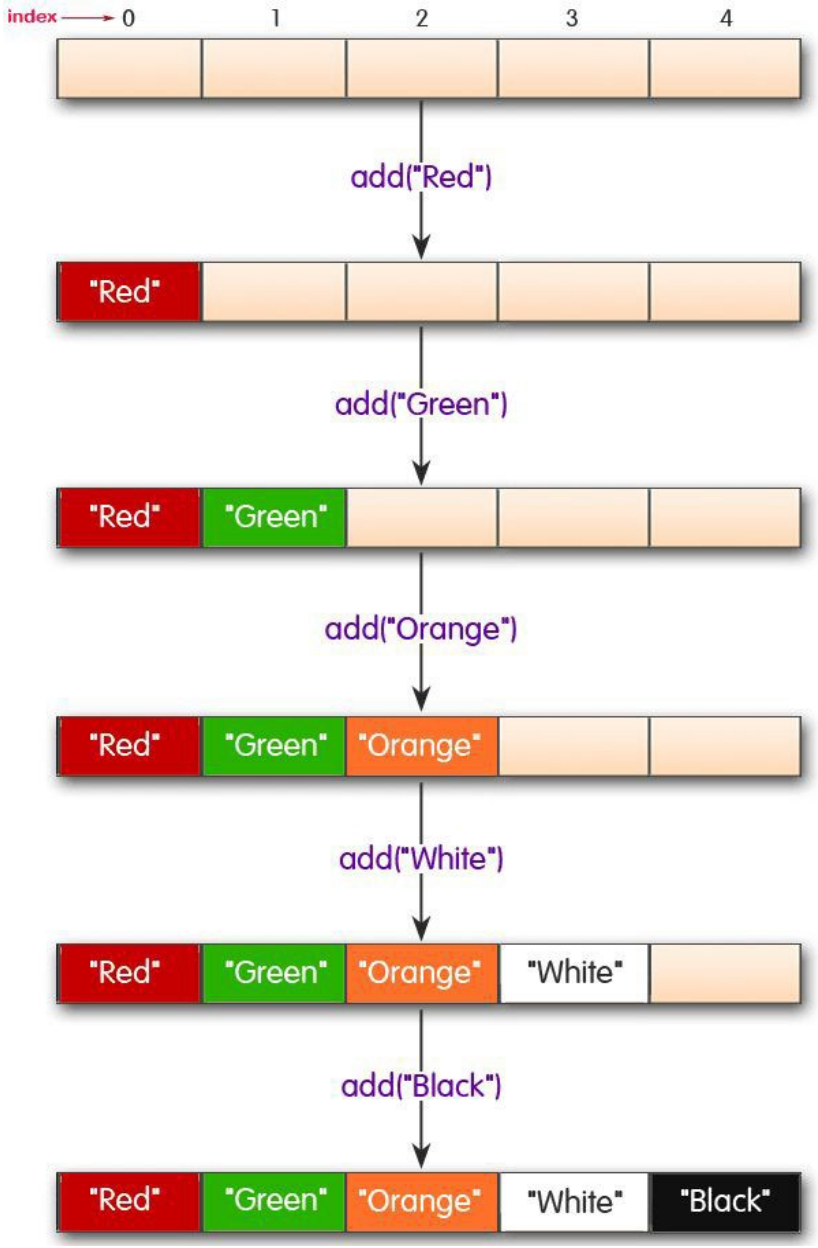
Ejercicios: ArrayList no genéricos (básicos) y genéricos

1. Manejo de un ArrayList no genérico (básico).

   - Crea un ArrayList **no genérico** de nombre *ejerc1* vacío.
   - Ahora, ve añadiendo los siguientes colores (que son en realidad objetos String con los nombres en inglés). Mira la figura:



Ejercicios: ArrayList no genéricos (básicos) y genéricos

- Imprime el contenido del ArrayList.
- Sobre el ArrayList del apartado anterior añade la cadena “Pink” en la posición 0. Muestra el contenido para comprobar.
- Ahora cambia la posición 3 actual, que contiene “Orange” y pon en su lugar “Blue”.
- Añade por el final el elemento “Purple”. Muestra el contenido para comprobar.
- Lee específicamente lo que hay en las posiciones pares.
- Elimina el color que exista en la posición 4. Muestra el contenido para comprobar.
- Finalmente, usando el método `Collections.sort(...)` vas a ordenar alfabéticamente los nombres de los colores.
- Muestra el contenido del array, después de ordenar, para comprobar.
- Añade un entero cualquiera. Muestra el contenido para comprobar.
- Muestra en pantalla la longitud de la cadena en la posición 3 del ArrayList.

2. Repite el ejercicio 1 pero ahora con un ArrayList con nombre `ejerc1Gen`, y asegúrate que sólo pueda contener Strings.

3. Crea la clase Perro:

```
class Perro {

   private String nombre;

   Perro(String n) {
      nombre = n;
   }
```

Ejercicios: ArrayList no genéricos (básicos) y genéricos

```
 public void ladra() {

     System.out.println(nombre + ": guau!");
 }

 public String toString() {
   return "soy el perro " + nombre;
 }
}
```

Crea la clase Gato:

```
class Gato {


   private String nombre;


   Gato(String n) {
        nombre = n;
   }

   public void maulla() { System.out.println(nombre +
        ": miau!");
   }

   @Override
   public String toString() {
        return "soy el gato " + nombre;
   }
}
```

Ejercicios: ArrayList no genéricos (básicos) y genéricos

Ahora, haz una clase Ejercicio3, con un método principal que realice estas acciones:

- Crea un ArrayList básico llamado `todos`, e intenta añadir los perros “Bobby”, “Jimmy”, “Oasis” y “Sanson”, y los gatos “Garfield”, “ConBotas”, “DGato”.
- Recorre ese ArrayList básico con un for mejorado (for-each) , y hazlos maullar o ladrar, según corresponda, pero sin tener que conocer a priori en qué índice hay un gato o un perro.
- Crea un ArrayList que sólo contenga perros, e intenta añadir “Bobby”, “Jimmy”, “Oasis” y “Sanson”, y los gatos “Garfield”, “ConBotas”, “DGato”.

4. Ahora vas a crear la clase Mascota con estas características:

    - Atributo nombre (String)
    - Constructor que recibe el nombre

Vas a crear una clase `PerroH` (de Perro en Herencia) que será hija de Mascota, y que tendrá un comportamiento similar a `Perro`. Es importante que no tenga un atributo nombre, porque ya lo tiene por herencia de Mascota, y su constructor ahora debería llamar al de la clase madre. De manera análoga, crea la clase `GatoH`.

Ahora, haz una clase Ejercicio4, con un método principal que realice estas acciones:

- Haz un ArrayList `misMascotas` que contenga solo objetos de la clase Mascota.
- Intenta añadir los perros (esta vez de la clase PerroH) “Bobby”, “Jimmy”, “Oasis” y “Sanson”, y los gatos (esta vez de la clase GatoH) “Garfield”, “ConBotas”, “DGato”.
- Recorre ese ArrayList genérico con un for mejorado, y hazlos maullar o ladrar según corresponda, pero sin tener que conocer a priori en qué índice hay un gato o un perro.

Ejercicios: ArrayList no genéricos (básicos) y genéricos

- Añade en Mascota un método cualquiera que imprima algo por pantalla, y vuelve a recorrer ese ArrayList genérico llamando a ese método para todos los objetos del ArrayList. Encuentra la diferencia en funcionamiento entre este punto y el anterior.

5. Uso de comodines o wildcards

- Escribe dentro de la clase Ejercicio4 un método estático `void procesaElementos1(ArrayList<? extends Mascota>)` , que lo que haga sea llamar al método que tú mismo añadiste en el último punto del ejercicio 4.
- En el Main crea un ArrayList de PerroH, y añade cuatro perros con nombres distintos, intenta llamar al método `procesaElementos1` con él como argumento
- En el Main crea un ArrayList de GatoH, y añade tres gatos con nombres distintos, intenta llamar al método `procesaElementos1` con él como argumento.
- En el Main crea un ArrayList de String, y añade los nombres de tus tres colores favoritos, e intenta llamar al método `procesaElementos1` con él como argumento.
- En el Main crea otro ArrayList, ahora sin tipado (básico), y añade los nombres de tus tres colores favoritos, e intenta llamar al método `procesaElementos1` con él como argumento.
- Averigua el significado de **`? super Nombre_de_clase`**
- Haz un método estático que sea `void procesaElementos2(ArrayList<? super Mascota>)` que haga las mismas acciones de `procesaElementos1`. Mira con qué tipo de ArrayList lo podrías llamar, y créalo (si no lo tienes) para comprobar que la llamada funciona.

6. Uso de Iteradores:

- Recorre el ArrayList `misMascotas` con un **iterador básico** (sin tipado) y muestra el nombre de todas ellas una a una.
- Recorre el ArrayList `misMascotas` con un **iterador genérico** (con tipado) y muestra el nombre de todas ellas una a una.
- Crea un ArrayList de String que se llamará `nombres`. Con un for mejorado, añade en él los nombres de todas las mascotas guardadas en misMascotas.

Ejercicios: ArrayList no genéricos (básicos) y genéricos

- Ahora, empleando un iterador genérico, recorre el ArrayList `nombres`, y borra (con el iterador directamente) aquellas cadenas cuya longitud sea diferente de 5.
- Comprueba que `nombres` contiene los valores que debería.
- Haz lo mismo que has hecho, pero sin usar iteradores.