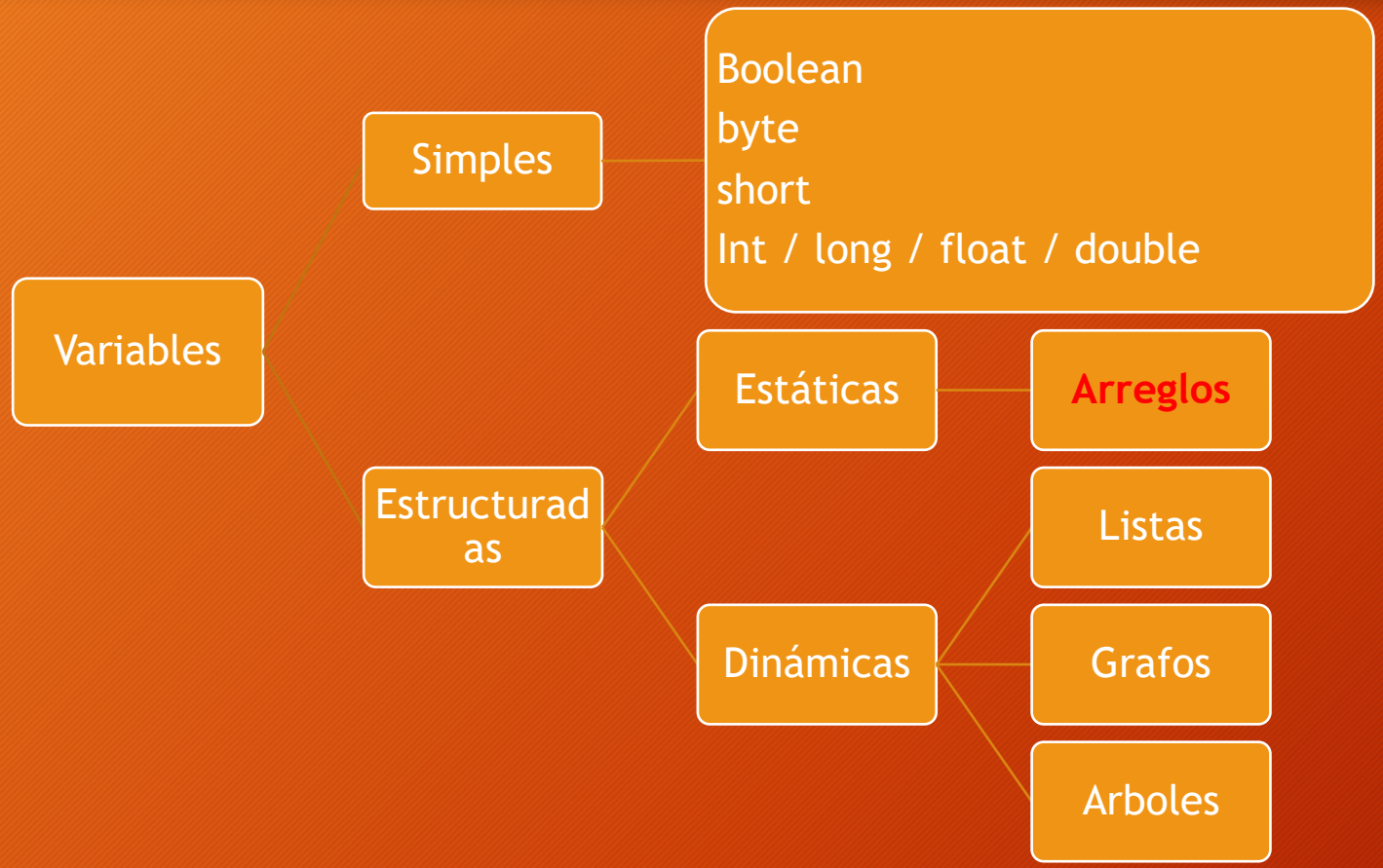
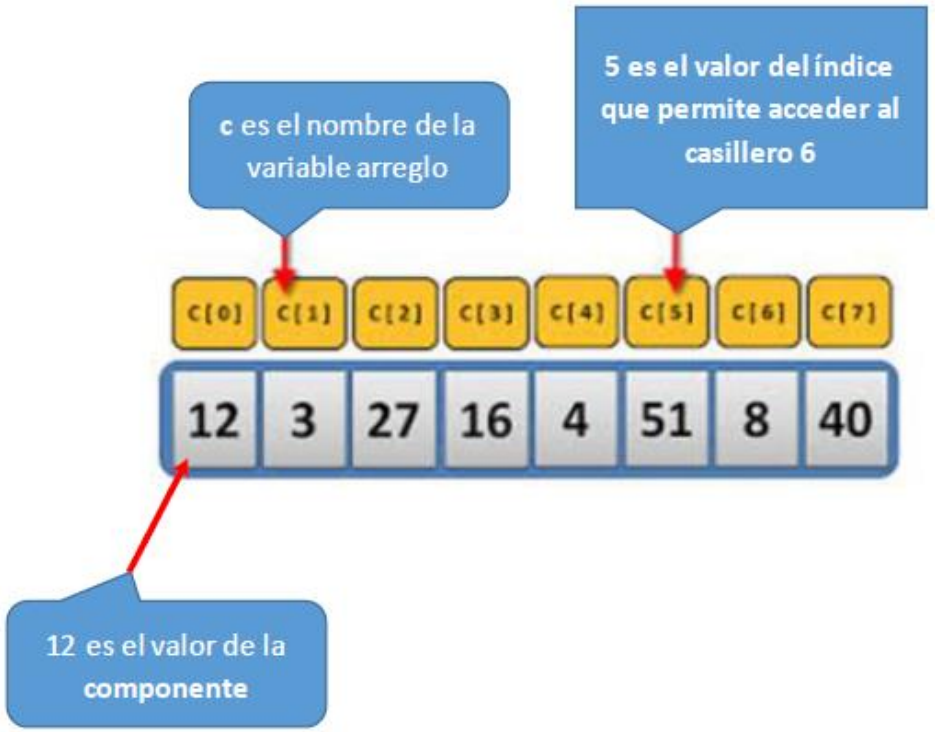
Estructuras de datos / Arreglos



Una de las estructuras más conocidas en el ámbito de la programación son los **arreglos unidimensionales.** Un arreglo (array) o vector es un conjunto de datos, todos del mismo tipo, almacenado secuencialmente en memoria. Cada valor o componente individual es asociado con un número entero llamado **índice.**

En **Java**, los arreglos son estructuras estáticas cuya declaración es la siguiente:

int vector [];



En **Java**:

- **int** c [] = **new** int[8];

```
c[0] = 12;
c[1] = 3;
c[2] = 27;
c[3] = 16;
c[4] = 4;
c[5] = 51;
c[6] = 8;
c[7] = 40;
```

- int [] c = {12, 3, 27, 16, 4, 51, 8, 40};

Para conocer el largo del vector(o tamaño) los vectores tienen el atributo **length**

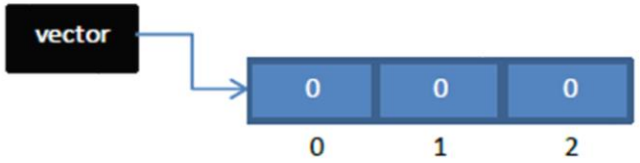
En **Java**, los arreglos son estructuras estáticas cuya declaración es la siguiente:

- Esa declaración es totalmente equivalente **int [] vector**. Es decir los corchetes pueden colocarse delante o detrás del identificador de variable.
- Técnicamente la declaración anterior se conoce como una referencia a un vector, que Java inicializará automáticamente en **null.**

- Para pedir memoria para las componentes es necesario utilizar el operador **new:**

*vector = **new** int[3];*

- Gráficamente lo anterior:



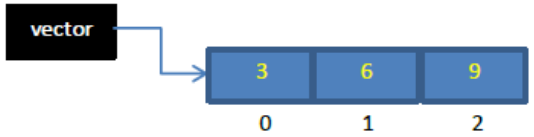
- Es posible de declarar y dimensionar la variable en la misma instrucción:

int vector[] = new int[3];

- Algunas consideraciones importantes sobre los vectores:
  - ✓ al ser estáticos, su tamaño queda definido en tiempo de compilación. Durante la ejecución del programa no es posible modificar su dimensión.
  - ✓ Notar que si el vector tiene 3 componentes la última casilla tiene índice **2** debido a que los índices siempre comienzan en 0.
  - ✓ Es posible *declarar*, dimensionar e inicializar sus componentes en una sola línea:

int vector[] = {3,6,9};

Resultando:



- Una vez creado el arreglo es posible conocer su cantidad de componentes o largo mediante:

vector.length; //cantidad de casillas

- Para acceder a cada casilla o componente:

*vector[**índice**]*

Por ejemplo para mostrar el contenido de la primera posición:

System.out.println(vector[0]);

Para modificar el valor de la última posición:

vector[2] = 10;

</digitalmind>

Algoritmos básicos

Las operaciones básicas más comunes que necesitamos aprender para manejar arreglos son:

- Recorrido
- Búsqueda
- Ordenamiento

Programación Orientada a Objetos

</digitalmind>

Java es un lenguaje fuertemente tipado. Esto significa que a todo recurso que vayamos a utilizar previamente le debemos definir su tipo de datos.

Definición 1: llamamos "**objeto**" a toda variable cuyo tipo de datos es una "clase".

Definición 2: llamamos "**clase**" a una estructura que agrupa datos junto con la funcionalidad necesaria para manipular dichos datos.

Sin saberlo, durante el capítulo anterior trabajamos con objetos. Las cadenas de caracteres son objetos de la clase `String`, por lo tanto, almacenan información (la cadena en sí misma) y la funcionalidad necesaria para manipularla.

Por ejemplo:

```
String s = "Hola Mundo";
int i = s.indexOf("M");
```

El objeto `s` almacena la cadena `"Hola Mundo"` y provee la funcionalidad necesaria para informar la posición de la primera ocurrencia de un determinado carácter dentro de dicha cadena.

El objeto **s** almacena la cadena "Hola Mundo" y provee la funcionalidad necesaria
para informar la posición de la primera ocurrencia de un determinado carácter dentro de dicha cadena.

Clases y objetos

</digitalmind>

Definición de Clase:

Que es una Clase? es como una categoría o un resultado de similitudes con precisión. Se puede decir que una clase es un prototipo que define propiedades y métodos comunes a múltiples objetos de un mismo tipo.

Como regla las clases de definen en singular usando el método CamelCase.

```
public class Fecha
{
   private int dia;
   private int mes;
   private int anio;
}
```

- Nombre de clase
- Atributos
- Métodos / Comportamientos

Definición Métodos:

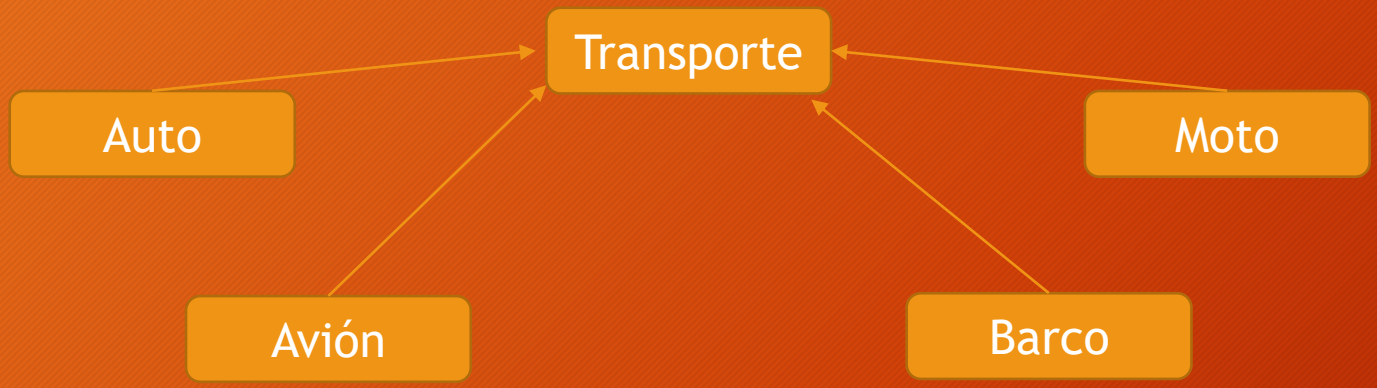
Los métodos de una clase se escriben como funciones. Dentro de los métodos, podemos acceder a los atributos de la clase como si fueran variables globales.

```
public class Fecha
{
   private int dia;
   private int mes;
   private int anio;

   public int getDia()
   {
      // retorna el valor de la variable dia
      return dia;
   }

   public void setDia(int dia)
   {
      // asigna el valor del parametro a la variable dia
      this.dia = dia;
   }
```

Ejemplos de Clases



</digitalmind>

En las clases se definen sus siguientes aspectos:

- **Atributos.** Es decir, los datos miembros de una clase. Estos datos pueden ser públicos (accesibles desde otra clase) o privados (solo accesibles por código de su propia clase). También se las llama "campos".
- **Métodos.** Es decir, las funciones miembros de la clase. Son las acciones (u operaciones) que puede realizar la clase.
- **Código de inicialización.** Para crear una clase, hace falta realizar normalmente operaciones previas. Esto es conocido como "el constructor de la clase".

Definición Objetos:

Una clase es un conjunto de objetos, por la tanto, un OBJETO pertenece a una clase, que además es una copia de una clase con todas sus características y rasgos similares.

Definición Herencia:

La herencia permite crear una clase general primaria o superclase, para luego crear clases mas especializadas que reutilizan el código de clase general.

```python
class Persona:
    nombres = ""
    apellidos = ""
    edad = 0
    direccion = ""
    def caminar(self):
        print("Está caminando...")
    def hablar(self):
        print("hablando...")

if __name__ == '__main__':
    persona = Persona()
    #persona.caminar()
    #persona.hablar()
    persona.nombres="Jorge"
    persona.apellidos = "Salamanca"
    persona.edad = 35
    persona.direccion = "La Heras 951 - Bogota - Colombia"
    print(persona.nombres,persona.apellidos, persona.edad, persona.direccion, "Se encuentra: ")
    persona.hablar(),
    print("Mientras: ")
    persona.caminar()
```