Asignatura 824
Programación Intermedia
Tema II

Cátedra Tecnología de Sistemas, 2023

Arreglo

Son estructuras de datos que consisten de elementos de datos relacionados del mismo tipo.

Los arreglos hacen que sea conveniente procesar grupos de valores relacionados. Estos arreglos conservan la misma longitud una vez que son creados.

Los arreglos son objetos, por ello son considerados tipos de referencia.

Los tipos primitivos frente a los tipos de referencia

Una variable de tipo primitivo puede contener a la vez exactamente un valor de su tipo declarado.

Variables de tipos de referencia (normalmente llamadas referencias) se utilizan para almacenar las ubicaciones de los objetos. Se dice que dicha variable se refiere a un objeto en el programa.

UNED

Declaración y creación de arreglos

La siguiente instrucción y expresión crea un objeto arreglo que contiene 12 elementos int y almacena la referencia del arreglo en la variable llamada c:

Forma 1:

```
int[] c = new int[12];
```

Forma 2:

```
int[] c; // declara la variable arreglo
c = new int[12]; // crea el arreglo; lo asigna a la variable tipo arreglo
```

Listas de parámetros de longitud variable

Se pueden crear métodos que reciben un número no especificado de parámetros.

```
// calcula el promedio
public static double promedio (double... numeros) {
   double total = 0.0;

   // calcula el total utilizando la instrucción for mejorada
   for (double d : numeros) {
      total += d;
   }

   return total / numeros.length;
}
```

```
public static void main(String[] args) {
   double d1 = 10.0;
   double d2 = 20.0;
   double d3 = 30.0;
   double d4 = 40.0;

   System.out.printf("d1 = %.1f%nd2 = %.1f%nd3 = %.1f%nd4 = %.1f%n%n",
      d1, d2, d3, d4);

   System.out.printf("El promedio de d1 y d2 es %.1f%n",
      promedio(d1, d2));
   System.out.printf("El promedio de d1, d2 y d3 es %.1f%n",
      promedio(d1, d2, d3));
   System.out.printf("El promedio de d1, d2, d3 y d4 es %.1f%n",
      promedio(d1, d2, d3, d4));
}
```

Manejo de excepciones

Una excepción indica un problema que se produce mientras se ejecuta un programa. El manejo de excepciones nos ayuda a crear programas tolerantes a errores que pueden resolver (o manejar) excepciones.

Cuando la JVM o un método detectan un problema, como un índice de arreglo no válido o un argumento de método no válido, lanza una excepción, es decir, se produce una excepción.

Manejo de excepciones

La instrucción “try”: Se utiliza para crear un bloque donde se puede generar una excepción.

La instrucción “catch”: permite crear un bloque que se ejecuta cuando se desencadena una excepción.

Instrucción “for” mejorada

Itera a través de los elementos de un arreglo sin utilizar un contador.

Sintaxis:

```
for (parámetro : nombreArreglo) {
   instrucción
}
```

Ejemplo:

```
// suma el valor de cada elemento al total
for (int numero : arreglo) {
   total += numero;
}
```

La clase “Arrays”

Brinda métodos “static” para manipulaciones comunes de arreglos.

También proporciona métodos de alto nivel como:

- sort: para ordenar un arreglo.
- binarySearch: para buscar en un arreglo.
- equals: para comparar arreglos.
- fill: para colocar valores en un arreglo.

Colecciones

Se utilizan para almacenar grupos de objetos relacionados en la memoria. Estas clases proporcionan métodos eficientes que organizan, almacenan y recuperan los datos sin necesidad de conocer cómo son almacenados.

Colección “ArrayList”

Puede cambiar dinámicamente su tamaño para acomodar más elementos.

Los elementos deben ser un mismo tipo.

Colección “ArrayList”

Cuenta con los siguientes métodos:

Método	Descripción
`add`	Sobrecargado para agregar un elemento al *final* de un objeto `ArrayList` o en un índice específico en un `ArrayList`.
`clear`	Elimina todos los elementos del objeto `ArrayList`.
`contains`	Devuelve `true` si el objeto `ArrayList` contiene el elemento especificado; de lo contrario, devuelve `false`.
`get`	Devuelve el elemento en el índice especificado.
`indexOf`	Devuelve el índice de la primera ocurrencia del elemento especificado en el objeto `ArrayList`.
`remove`	Sobrecargado. Elimina la primera ocurrencia del valor especificado o del elemento en el índice especificado.
`size`	Devuelve el número de elementos almacenados en el objeto `ArrayList`.
`trimToSize`	Recorta la capacidad de `ArrayList` al número actual de elementos.

Introducción a las clases y a los objetos

Variables de instancia

Un objeto tiene atributos, implementados como variables de instancia y llevados consigo durante toda su vida útil.

Las variables de instancia existen antes de llamar a los métodos en un objeto, mientras que los métodos se ejecutan y después de que los métodos completan su ejecución. Cada objeto (instancia) de clase tiene su propia copia de las variables de instancia de clase.

Una clase normalmente contiene uno o más métodos que manipulan las variables de instancia pertenecientes a objetos particulares de la clase.

Set’s y get’s

Los atributos son privados, es decir, sólo en la misma clase se pueden “ver”.

Sólo se puede obtener y modificar los valores de los atributos por medio de los set’s y get’s.

Asignamos un valor a un atributo por medio del set.

Obtenemos el valor por medio del get.

UNED

Set's y get's

```
public class Cuenta {
   private String Cuenta; // variable de instancia

   // método para establecer el nombre en el objeto
   public void setNombre(String nombre) {
      this.nombre = nombre; // almacena el nombre
   }

   // método para recuperar el nombre del objeto
   public String getNombre() {
      return nombre; // devuelve el valor del nombre al método que llama
   }
}
```

Introducción a las clases y a los objetos

Constructor

- Permite la creación de objetos de una clase.
- Puede tener n cantidad de sobrecargas

Sintaxis: public NombreClase()

El método finalize

- Método para “destruir” objetos.

- No es necesario llamarlo.

Constructor

- Cuando no se declara un constructor, el compilador proporciona uno predeterminado.
- Pueden crearse las sobrecargas del constructor que se desee.
- Las variables de instancia se pueden inicializar (asignar un valor) desde el constructor.

Clases y objetos: un análisis más detallado

Control de acceso a los miembros

- Los modificadores de acceso público y privado controlan el acceso a las variables y métodos de una clase.
- El propósito principal de los métodos públicos es presentar a los clientes de la clase una vista de los servicios que proporciona (es decir, la interfaz pública de la clase).
- Los clientes no necesitan preocuparse por cómo la clase logra sus tareas. Por esta razón, las variables privadas de la clase y los métodos privados (es decir, sus detalles de implementación) no son accesibles para sus clientes.

Control de acceso a los miembros

```
public class PruebaAccesoMiembro {
   public static void main(String[] args) {
      Hora1 hora = new Hora1(); // crea e inicializa el objeto Hora1

      hora.hora = 7; // error: hora tiene acceso privado en Hora1
      hora.minuto = 15; // error: minuto tiene acceso privado en Hora1
      hora.segundo = 30; // error: segundo tiene acceso privado en Hora1
   }
}
```

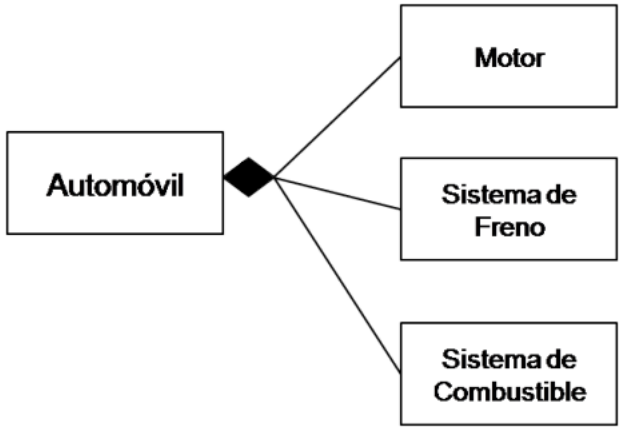
Referencias a los miembros del objeto actual con la referencia “this”

- Palabra reservada de java
- Hace referencia al objeto en la que estamos
- Sirve para diferenciar variables o parámetros de nuestros atributos
- Para acceder los métodos de nuestra clase

Clases y objetos: un análisis más detallado

Composición

- Una clase puede tener referencias a objetos de otras clases como miembros.
- Relación “tiene un”.



Variables final

- Tienen un valor que no se puede cambiar
- Si no se asigna un valor se presenta error de compilación
- Se les puede dar valor sólo en su declaración o en el constructor

“Si no estás dispuesto a aprender nadie te puede ayudar. Si estás dispuesto a aprender nadie te puede parar.” (Proverbio chino)

¡GRACIAS!