

The logo for Oracle Academy is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is framed by two horizontal dark gray bars, one at the top and one at the bottom.

ORACLE

Academy

Java Fundamentals

7-1:

Clases, objetos y métodos

ORACLE
Academy



Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

Objetivos

- Esta lección abarca los siguientes temas:
 - Reconocimiento del formato general correcto de una clase
 - Creación del objeto de una clase
 - Creación de métodos que se copilan sin errores
 - Devolución de un valor a partir de un método
 - Uso de parámetros en un método
 - Creación de una clase de controlador y suma de instancias de clases de objetos
 - Suma de un constructor a una clase
 - Aplicación del nuevo operador
 - Descripción de la recolección de elementos no utilizados y finalizadores
 - Aplicación de la referencia "this"
 - Suma de un constructor para inicializar un valor



Creación de la plantilla de una clase

- Los programadores pueden crear sus propias clases
- Las clases son esencialmente una plantilla o plano para todas las instancias de una clase
- El código de la clase también comunica al compilador cómo definir, crear e interactuar con los objetos de la clase
- El código de la siguiente diapositiva comienza a crear el vehículo de la clase que representaría el esquema básico para los objetos del vehículo

Ejemplo de la creación de la plantilla de una clase

```
public class Vehicle {  
    // the Vehicle class has two fields  
    private String make;  
    private int milesPerGallon;  
    //constructor  
    public Vehicle(){  
    } //end constructor  
    //mutator/setter methods  
    public void setMake(String m){  
        make = m;  
    } //end method setMake  
    public void setMilesPerGallon(int mpg){  
        milesPerGallon = mpg;  
    } //end method setMilesPerGallon  
    public String getMake(){  
        return make;  
    } //end method getMake  
    public int getMilesPerGallon(){  
        return milesPerGallon;  
    } //end method getMilesPerGallon  
} //end class Vehicle
```

Creación de una instancia de una clase

- Una vez que ha creado una clase, puede crear instancias de esta clase (objetos) en una clase de controlador o dentro de otras clases de objetos
- Instancias:
 - heredan todos los atributos y métodos definidos en la plantilla de una clase
 - Interactúan independientemente entre sí
 - Son objetos de referencia
 - Se crean utilizando el nuevo operador

Al igual que sucede con Strings, las referencias almacenan la dirección del objeto. (Por supuesto, Strings son objetos.)

Crear una instancia de la instancia

- Para crear una instancia de una instancia de un vehículo denominado myCar, escriba:

```
public class VehicleTester{  
    public static void main(String[] args){  
        Vehicle myCar = new Vehicle();  
    } //end method main  
} //end class VehicleTester
```

En Java, la creación de una instancia significa la creación de objetos de una clase.



Constructores

- Los constructores son métodos que permiten al usuario crear instancias de una clase (instanciar)
- En programación se aconseja incluir un constructor predeterminado en las clases
- Los constructores que contienen parámetros generalmente inicializan las variables privadas de la clase en los valores aprobados por el usuario
- Los constructores no tienen un tipo de devolución (anulados o de otro tipo)



Los constructores no tienen un tipo de retorno porque devuelven un nuevo objeto de esa clase.

Constructor predeterminado

- En programación se aconseja incluir a un constructor predeterminado en las clases
- Un constructor predeterminado:
 - No toma parámetros
 - En general, inicializa todas las variables privadas en los valores de base

```
public Vehicle() {  
    make = "";  
    milesPerGallon = 0;  
} //end constructor
```

Constructor con parámetros

- Un constructor con parámetros se utiliza cuando desea inicializar las variables privadas en valores distintos de los valores predeterminados

```
public Vehicle(String m, int mpg){  
    make=m;  
    milesPerGallon=mpg;  
} //end constructor
```

Parámetros

Los parámetros son variables que están enumeradas como parte de la declaración de un método (o constructor). En el ejemplo anterior, los String m e int mpg son parámetros. Se otorgan valores a los parámetros cuando se invoca el método o el constructor.



Creación de una instancia para una instancia Vehículo

- Para crear una instancia Vehículo utilizando el constructor con los parámetros, escriba:

Argumentos

```
Vehicle myCar = new Vehicle("Toyota", 30);
```

- Para crear una instancia Vehículo utilizando el constructor predeterminado, escriba:

```
Vehicle myCar = new Vehicle();
```



Definición de métodos

- Un método es un bloque de código al que se hace referencia mediante el nombre y que se puede invocar en cualquier punto del programa simplemente mediante el uso del nombre del método
- Hay cuatro elementos principales para definir su propio método:
 - Modificador de acceso (público, privado, protegido, predeterminado)
 - Tipo de devolución
 - Nombre del método
 - Parámetros

```
public returnType methodName(Parameter p, ...)  
{  
    /*code that will execute with each call to the method goes here*/  
}
```

El modificador de acceso por defecto se indica a través de la ausencia de público, privado o protegido.

No use la palabra "por defecto".

Por lo general, los métodos (y clases) tendrán el modificador de acceso público. Normalmente, los campos Non-constant tendrán el modificador de acceso privado.

Componentes de un método

- Los componentes de un método incluyen:
 - Tipo de devolución:
 - Identifica qué tipo de objeto será devuelto al invocar el método, si es que existe un objeto
 - Si no se devuelve nada, el tipo de devolución se declara nula
 - Nombre del método:
 - se utiliza para invocar el método

Componentes de un método

- Parámetros:
 - El programador puede decidir que se incluyan parámetros según la finalidad y la función del método
 - Los parámetros pueden ser de cualquier primitivo o tipo de objeto, pero el tipo de parámetro utilizado al invocar el método debe coincidir con el tipo de parámetro especificado en la definición del método

Ejemplo de componentes de un método

Tipo de
devolución

Nombre del
método

Parámetros

```
public String getName(String firstName, String lastName)
{
    return( firstName + " " + lastName );
} //end method getName
```

Métodos de clases

- Cada clase tendrá un conjunto de métodos asociados que permiten la funcionalidad de la clase
- Método de acceso
 - “captador”
 - Devuelve el valor de una variable privada específica
- Método de descriptores de mutación
 - “establecedor”
 - Cambia o establece el valor de una variable privada específica
- Método funcional
 - Devuelve o realiza un tipo de funcionalidad para la clase

Métodos de acceso

- Los métodos de acceso acceden y devuelven el valor de una variable privada específica de la clase
- El tipo de devolución no nula corresponde al tipo de datos de la variable a la que está accediendo
- Incluye una sentencia de devolución
- Generalmente, no tiene parámetros

```
public String getMake() {  
    return make;  
} //end method getMake  
  
public int getMilesPerGallon() {  
    return milesPerGallon;  
} //end method getMilesPerGallon
```



Métodos de descriptores de mutación

- Los métodos de descriptores de mutación establecen o modifican el valor de una variable privada específica de la clase
- Tipo de devolución nula
- Parámetro con un tipo que corresponde al tipo de variable establecida

```
public void setMake(String m) {\n    make = m;\n} //end method setMake\n\npublic void setMilesPerGallon(int mpg) {\n    milesPerGallon = mpg;\n} //end method setMilesPerGallon
```



ORACLE
Academy

JF 7-1
Clases, objetos y métodos

Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

18

Los métodos Mutator también pueden tener código adicional para probar o modificar los parámetros.

Por ejemplo, podría haber otro método setMilesPerGallon con un parámetro doble. El cuerpo del método podría añadir 0,5 y, a continuación, convertir el valor en un int y asignarlo al campo.

Métodos funcionales

- Los métodos funcionales realizan una funcionalidad para la clase
- Tipo de devolución nula o no nula
- Los parámetros son opcionales y se utilizan según lo que se requiera para la función del método



Métodos funcionales

- A continuación se muestra un método funcional para el vehículo de la clase que compara dos vehículos y devuelve un valor int (entero) para la comparación

```
//Compares the miles per gallon of each vehicle passed in,  
returns 0 if they are the same, 1 if the first vehicle is  
larger than the second and -1 if the second vehicle is  
larger than the first*/
```

```
public int compareTo(Vehicle v1, Vehicle v2){  
    if(v1.getMilesPerGallon() == v2.getMilesPerGallon())  
        return 0;  
    if(v1.getMilesPerGallon() > v2.getMilesPerGallon())  
        return 1;  
    return -1;  
} //end method compareTo
```

Ejemplo del uso de constructores y métodos en el método principal de la clase de controlador

- Para lo siguiente:
 - ¿Qué funcionalidad tiene cada línea?
 - ¿Qué imprimirá la sentencia final de impresión en la pantalla?

```
public class VehicleTester{  
    public static void main(String[] args){  
        Vehicle v;  
        v=new Vehicle();  
        v.setMake("Ford");  
        v.setMilesPerGallon(35);  
  
        System.out.print("My "+v.getMake()  
                        + " gets " + v.getMilesPerGallon()  
                        + " mpg.");  
    }  
}
```

Referencia "this"

- En un método de instancia o un constructor, this es una referencia al objeto actual
- La referencia al objeto cuyo método o constructor se invoca
- Consulte a cualquier miembro del objeto actual utilizando this
- Se utiliza mayormente cuando un campo está sombreado por un parámetro de un método o constructor del mismo nombre

Ejemplo de referencia "this"

- Cuando el argumento de un método “sombrea” el campo de un objeto, la referencia "this" se usa para diferenciar el alcance local del alcance de la clase

```
public class Point {  
    private int x;  
    Private int y;  
  
    //constructor  
    public Point(int x, int y) {  
        this.x = x;  
        this.y = y;  
    } //end constructor  
} //end class
```

Ejemplo de la clase Naipes

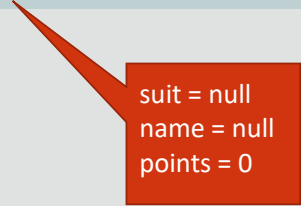
- Considere un juego de naipes estándar
- Para representar a cada naipе como la instancia de la clase Naipе, ¿qué atributos necesitaría tener la clase?
 - Palo
 - Nombre
 - Puntos

```
public class Card {  
    private String suit;  
    private String name;  
    private int points;  
} //end class Card
```


Representación de objetos de referencia

- Al crear una nueva instancia de un objeto, se hace referencia al objeto en la memoria
- La referencia apunta al objeto
- Todas las variables de los atributos se crean y se inicializan en base al constructor utilizado

```
Card c = new Card();
```



suit = null
name = null
points = 0

Ejemplo de comprensión de la recolección de elementos no utilizados

- Al considerar el código que figura a continuación, ¿qué ocurrirá en la memoria después de la línea `c2 = c;` ?
- Al ejecutarla, `c2 = c;` toma la referencia `c2` y hace referencia al mismo objeto como `c`
- De este modo, el objeto `c2` original queda efectivamente inutilizable y la recolección de elementos no utilizados se ocupa de este objeto eliminándolo de la memoria

```
Card c=new Card("Diamonds", "Four", 4);  
Card c2=new Card("Spades", "Ace", 1);  
c2 = c;
```



Finalizadores

- Un finalizador es un código invocado por el recolector de elementos no utilizados cuando determina que no existen más referencias al objeto
- Todos los objetos heredan un método `finalize()` de `java.lang.Object`
- Este método no toma parámetros y se escribe para que no realice ninguna acción al ser invocado

Finalizadores

- Al invalidar el método `finalize()` en las clases podrá modificar lo que ocurre antes de la recolección de elementos no utilizados, por ejemplo:
 - Notificar al usuario acerca de la recolección de elementos no utilizados que va a ocurrir
 - Limpiar los recursos que no sean de Java, por ejemplo, cerrar un archivo

Ejemplo del método de finalización

- Este es un ejemplo del método `finalize()` invalidado en una clase
- Cierra todos los archivos asociados y notifica al usuario cuando se produce la finalización

```
protected void finalize(){
    try{
        close(); //close all files
    }//end try
    finally{
        System.out.println("Finalization has occurred");
    }//end finally
} //end method finalize
```

Terminología

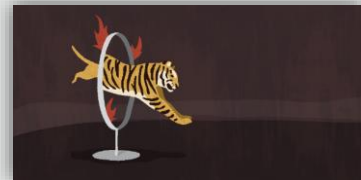
- Los términos clave usados en esta lección son los siguientes:
 - Método de acceso
 - Clase
 - Constructor
 - Finalizadores
 - Recolección de elementos no utilizados
 - Inicialización
 - Creación de una instancia
 - Método

Terminología

- Los términos clave usados en esta lección son los siguientes:
 - Método de descriptores de mutación
 - Nuevo
 - Nulo
 - Objeto
 - Referencia
 - Referencia "this"

Resumen

- En esta lección, habrá aprendido a:
 - Reconocer el formato general correcto de una clase
 - Crear el objeto de una clase
 - Crear métodos que se copilan sin errores
 - Devolver un valor a partir de un método
 - Usar parámetros en un método
 - Crear una clase de controlador y sumar instancias de clases de objetos
 - Sumar un constructor a una clase
 - Aplicar el nuevo operador
 - Describir la recolección de elementos no utilizados y finalizadores
 - Aplicar la referencia "this"
 - Sumar un constructor para inicializar un valor



The Oracle Academy logo is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is framed by a thin black border, with dark gray horizontal bars at the top and bottom.

ORACLE

Academy