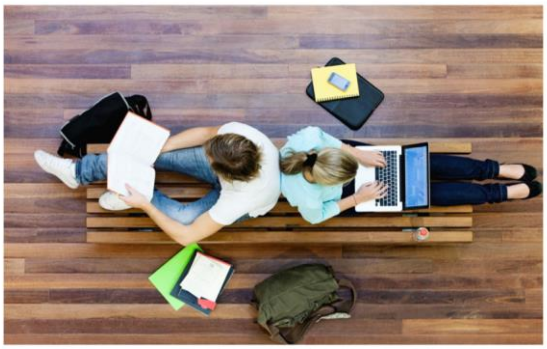

ORACLE
Academy

Java Fundamentals

7-2:

Parámetros y métodos de sobrecarga



ORACLE
Academy

Objetivos

- Esta lección abarca los siguientes temas:
  - Uso de modificadores de acceso
  - Paso de objetos a métodos
  - Devolución de objetos desde los métodos
  - Uso de métodos de argumentos variables
  - Sobrecarga de constructores
  - Sobrecarga de métodos
  - Escritura de una clase con arreglas específicas, constructores y métodos

Modificadores de acceso

- Los modificadores de acceso especifican el acceso para cambiar variables, métodos y clases
- Hay cuatro modificadores de acceso en Java:

Modificador de acceso	Descripción
public (Público)	Permite el acceso desde cualquier lugar
protected (Protegido)	Permite el acceso únicamente dentro del paquete que contiene el modificador
Private (Privado)	Únicamente permite el acceso desde adentro de la misma clase
"default" (Predeterminado - no especificado/en blanco)	Permite el acceso dentro de la clase, subclase u otras clases del mismo paquete que el modificador

Modificador de acceso público

- Los modificadores de acceso público permiten el acceso desde cualquier lugar
- En Java, al agregar la palabra clave público cuando se declara la variable, el método o la clase, convierte a la variable, el método o la clase en accesibles desde cualquier lugar

Las clases y métodos suelen ser de carácter público.

Declaración como Público

- El siguiente código muestra de qué manera se declaran públicos a una variable, un método o una clase
  - Variable:

```
public int milesRan = 2;//public access
int timePassed = 17;//access not specified
```

  - Método:

```
public int addMiles(int a, int b){
   return a+b;
}//end method addMiles
```

  - Clase:

```
public class Jogging{
   //class code here
}//end class Jogging
```


Las variables locales no tienen un modificador de acceso. Solo las variables de instancia utilizan modificadores de acceso.

Modificadores de acceso protegidos y predeterminados

- El modificador de acceso protegido permite el acceso dentro de la clase, subclase u otras clases del mismo paquete que el modificador
- Para declarar protegidos a una variable, un método o una clase, escriba la palabra clave protegido en lugar de público
- Un modificador de acceso predeterminado permite el acceso desde adentro del mismo paquete
- Para declarar como predeterminado a una variable, un método o una clase, debe incluir un modificador de acceso

Un error común que suelen cometer los nuevos programadores de Java es que no especifican ningún modificador de acceso y, por lo tanto, obtienen el que se usa por defecto.

Modificador de acceso privado

- Modificador de acceso privado:
  - solo permite el acceso desde adentro de la misma clase
  - Es el modificador de acceso más restrictivo
  - Es lo contrario del modificador de acceso público

```
private int bankAccountNumber;
```

Los campos suelen ser de carácter privado.

Cuando utilizar Público o Privado

Tipo	Definición	Cuándo se usa
`public` (Público)	Permite el acceso desde cualquier lugar	Cuando no importa si cualquier persona puede tener acceso a su código o desea compartir su código con otros
`private` (Privado)	Solo permite el acceso desde adentro de la misma clase	Cuando es importante que su código sea seguro y que no se pueda tener acceso a este desde cualquier lugar, excepto dentro de la misma clase

Objetos como parámetros

- Un parámetro es una variable en una declaración de método que se pasa al método

```
public int method(int parameter1, int parameter2)
```

- Los tipos de parámetros son aquellos que se pueden pasar a un método
- Esto incluye:
  - Tipos primitivos (int (entero), doble, char (carácter))
  - Objetos
    - String
    - arreglo

```
public int method(int anInt, double aDouble,
                  String aString, MyClassName anObjectOfMyClass)
```


No diferencia si el objeto procede de una clase API o si procede de una clase creada por el usuario.

Ejemplo de objetos como parámetros

- Un empleador tiene una vacante para promover a uno de sus empleados
- Desea crear un método que tome a un empleado como parámetro y calcule y devuelva las calificaciones de los empleados en base a sus competencias para la nueva posición

```
public int promotion(Employee E){
   int timeEmployed = E.getLengthOfEmployment();
   //do some calculations to set a rating for E
   return rating;
}//end method promotion
```

Paso de objetos como parámetros

- El paso de objetos como parámetros permite un acceso mucho más sencillo a la información que contiene el objeto
- Además, permite realizar cambios a los objetos dentro de un método y, también, permite comparar dos objetos que no pueden usar métodos de comparación primitivos

Devolución de objetos

- La escritura del método que devuelve un objeto es muy similar a la escritura del método que devuelve un tipo primitivo
- Por ejemplo, el empleador del ejemplo anterior acaba de aprender que los métodos pueden devolver un objeto
- Para poder encontrar a un empleado a ser promocionado de manera más sencilla puede escribir un método que tome a dos empleados
- El método devuelve al empleado que tiene la mejor calificación
- Este método es más sencillo que realizar el análisis de cada empleado, recuperar las calificaciones de cada uno y luego compararlo

Ejemplo de devolución de objetos

- El empleado identifica qué se devuelve

Para devolver un objeto, simplemente escriba el tipo de objeto aquí

```
public Employee promotion(Employee A, Employee B){
   //calculate to compare which employee is better
   //if employee A is better
   return A;
   //if employee B is better
   return B;
}//end method promotion
```

Métodos de argumentos variables

- Un método de argumentos variables:
  - es un método escrito para manejar una cantidad variable de argumentos
  - Solo funciona si invoca al método con el mismo tipo de argumento que el método requiere
- El método de argumento variable se verá de la siguiente manera:

```
public int total(int ... nums){
   int sum = 0;
   for(int i = 0; i < nums.length; i++)
      sum += nums[i];
   return sum;
}//end method total
```

Ejemplo de métodos de argumentos variables

- Por ejemplo, un método inicializado con un argumento variable de enteros no puede ser invocado con una cantidad no determinada de Strings, pero puede ser invocado únicamente con una cantidad de enteros para el argumento
- Si se declara otro método con un argumento variable de Strings, estas deben invocar a ese método con los String(s) que cumplan con los argumentos

¿Por qué se utilizan las arreglas en métodos de argumentos variables?

- ¿Por qué no usar solo una arreglo?
  - En un programa, debe conocer la cantidad de elementos de una arreglo para crear una
  - Si la cantidad de elementos cambia, necesitará una arreglo diferente para cada longitud diferente
  - El uso de un argumento variable permite el uso del método, sin tener siquiera que inicializar una arreglo
  - Además, permite varios usos con una cantidad variable de elementos

Métodos de argumentos variables y enteros

- ¿El método de argumentos variable solo funciona con enteros?
  - No, el argumento variable funciona con cualquier tipo primitivo, objeto e incluso con arreglas
  - Usted puede tener un argumento variable de arreglas

Ejemplo de empleado

- Para determinar las promociones de los empleados, el empleador codificó un método que comparó a dos empleados y devolvió al mejor
  - Ahora que el empleador cuenta con el método para comparar a los empleados, necesita un sistema para comparar a todos los empleados al mismo tiempo en lugar de comparar a dos empleados a la vez
  - En este caso, los argumentos variables pueden resultar prácticos

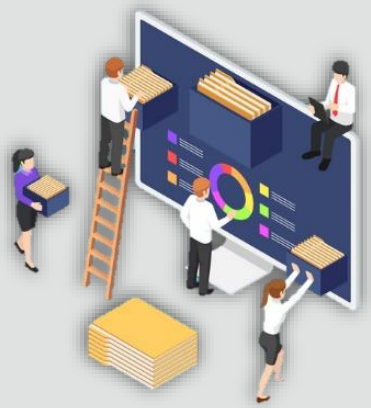
Ejemplo de código de argumentos variables de empleados

- Código para comparar a todos los empleados:

```
public Employee promotion(Employee ... employees){
   Employee bestCandidate = employees[0];
   //go through the list of employees and calculate
   //which one is the best candidate
   for(int i = 1; i < employees.length; i++){
     //if there is a candidate better than the current best
     if(employees[i].getRating > bestCandidate.getRating){
        //update the bestCandidate to the better one
        bestCandidate = employees[i];
     }//end if
   }//end for
   //return the best candidate found for the promotion
   return bestCandidate;
}//end method promotion
```

Invocación de un método con argumentos variables

- La invocación de un método con argumentos variables es como la invocación de cualquier otro método
- Sin embargo, se puede invocar con una cantidad diferente de argumentos cada vez que se invoca



ORACLE Academy

Invocación de un método con argumentos variables

- El código a continuación demuestra esta idea
- Sam, Érica, Dominic, Sandy y Jake son empleados
- El empleador intenta promocionar a Sam, Érica o Dominic como gerente y a Sandy o Jake como subgerente

```
//This compares Sam, Erica, and Dominic and assigns
//the best candidate of the 3 to newManager.
Employee newManager = promotion(sam, erica, dominic);

//This compares Sandy and Jake and assigns the better
//of the 2 to newAssistantManager
Employee newAssistantManager = promotion(sandy, jake);
```

Sobrecarga de constructores

- Los constructores asignan valores iniciales a las variables de instancia de una clase
- Los constructores dentro de una clase se declaran como métodos
- La sobrecarga de un constructor significa tener más de un constructor con el mismo nombre pero diferentes tipos y/o cantidad de argumentos

Ejemplo 1 de sobrecarga de constructores

- Este ejemplo sobrecarga al constructor público de la clase Perro

```
public class Dog{
   public Dog(){
      ...implementation...
   }//end constructor
   public Dog(int weight){
      ...implementation...
   }//end constrauctor
   public Dog(String barkNoise){
      ...implementation...
   }//end constructor
   public Dog(int weight, int loudness, String barkNoise){
      ...implementation...
   }//end constructor
}//end class Dog
```

Cómo funciona la sobrecarga de constructores

- La sobrecarga de constructores funciona de la siguiente manera:
  - Java lee al constructor en base a los argumentos que se le pasen
  - Una vez que identifica el nombre del constructor, compara los tipos de argumentos
  - Si los tipos de argumentos no coinciden con el primer constructor de ese nombre, continuará con el segundo y el tercero y así sucesivamente hasta identificar que el nombre del constructor y el tipo de argumento coinciden
  - Si no encuentra una coincidencia, el programa no se compilará

Ejemplo 2 de sobrecarga de constructores

```
public class Dog{
   private int weight;
   private int age;
   private String barkNoise;

   public Dog(){
     weight = 12;
     loudness = 4;
     barkNoise = "Woof";
   }//end constructor

   public Dog(int w, int l){
     weight = w;
     loudness = l;
     barkNoise = "ARF!";
   }//end constructor

   public Dog(int w, int l, String bark){
     weight = w;
     loudness = l;
     barkNoise = bark;
   }//end constructor
}//end class Dog
```

Este es un constructor que especifica el peso del perro y el sonido de los argumentos

Este es un constructor que especifica el peso del perro, el sonido y los ladridos en los argumentos

Explicación del ejemplo 2 de sobrecarga de constructores

- Dog() es el constructor predeterminado
- Un constructor predeterminado no tiene argumentos
- Si inicializó a un objeto Perro utilizando este constructor, tendrá 12 como el peso, 4 como el sonido y “wuf” como ladridos
- Los últimos dos constructores en la clase Perro permiten que difiera la asignación de variables de instancias de acuerdo con las especificaciones realizadas durante la inicialización

Explicación del ejemplo 2 de sobrecarga de constructores

- Aunque el constructor predeterminado Perro tenga el código para inicializar las variables de la clase, el código es opcional para un constructor predeterminado
- Si el constructor predeterminado no tiene código, las variables de la clase se inicializan con:
  - nulo para objetos
  - 0 (cero) para tipos numéricos primitivos
  - falso para booleanos

Explicación del ejemplo 2 de sobrecarga de constructores

- Si un constructor no se escribe para una clase, el constructor predeterminado (sin código) recibe el JVM
- Si no se escribe el constructor predeterminado y existen uno o más constructores adicionales, JVM no entregará el constructor predeterminado

Tome nota del segundo punto, ya que se suele pasa por alto.

Sobrecarga de métodos

- Al igual que la sobrecarga de constructores, la sobrecarga de un método tiene lugar cuando el tipo y/o la cantidad de parámetros difieren
  - A continuación se muestra un ejemplo de una situación donde el método necesitará una sobrecarga
    - Cree la clase Perro, luego cree la instancia de Perro en la clase de un controlador
    - Invoque (use) ambos métodos bark()

```
public class Dog{
   private int weight;
   private int loudness;
   private String barkNoise;

   public void bark(String b){
         System.out.println(b);
   }//end method bark
   public void bark(){
         System.out.println("Woof");
   }//end method bark
}//end class Dog
```

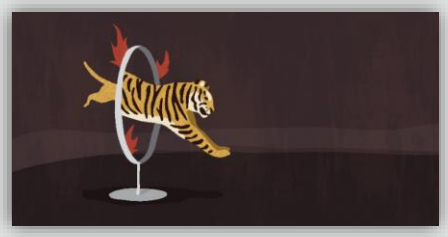

Un error común es intentar sobrecargar un método cambiando solo el tipo de retorno. La sobrecarga de métodos solo implica el cambio de los tipos de parámetros y el número de parámetros.

Terminología

- Los términos clave usados en esta lección son los siguientes:
  - Modificador de acceso
  - Constructor
  - Constructor predeterminado
  - Sobrecarga
  - Modificador de acceso privado
  - Modificador de acceso público
  - Método de argumentos variables

Resumen

- En esta lección, habrá aprendido a:
  - Usar modificadores de acceso
  - Pasar objetos a métodos
  - Devolver objetos desde los métodos
  - Usar métodos de argumentos variables
  - Sobrecargar constructores
  - Sobrecargar métodos
  - Escribir una clase con arreglas específicas, constructores y métodos




ORACLE
Academy