ORACLE

Academy

Java Foundations

7-6

Variables y métodos estáticos

ORACLE
Academy

Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Describir una variable estática y demostrar su aplicación dentro de un programa
  - Describir un método estático y mostrar su uso dentro de un programa
  - Comprender cómo se utiliza la palabra clave final con las variables estáticas

Revisión de las referencias de objeto

- Un objeto se debe instanciar para que se pueda acceder a sus campos y métodos
- La instanciación nos proporciona una referencia de objeto
- Una referencia de objeto se utiliza para acceder a los campos y métodos de un objeto

```
Prisoner p01 = new Prisoner()
p01.name                    //Accessing a field
p01.display()               //Calling a method
```

La clase Math es diferente

- Sería tedioso crear un nuevo objeto Math cada vez que deseamos realizar alguna operación matemática
- Afortunadamente, nunca es necesario instanciar un objeto Math
- Se puede acceder a los campos y métodos de Math directamente haciendo referencia a la clase Math
- Se denominan variables estáticas y métodos estáticos

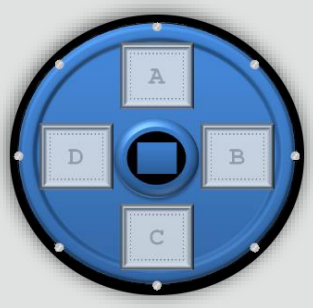
```
                          //Nothing instantiated
Math.PI                   //Accessing a static field
Math.sin(0)               //Calling a static method
```

¿Qué significa?

- ¿Por qué son importantes estos dos hechos?
  - Una referencia de objeto se utiliza para acceder a los campos y métodos de un objeto
  - A los campos y métodos estáticos se accede directamente haciendo referencia a la clase
- Es algo más que la comodidad de no tener que instanciar un objeto
- El siguiente ejercicio le permite explorar un caso de uso de datos estáticos
  - A continuación, proporcionaremos un resumen de lo que puede haber advertido

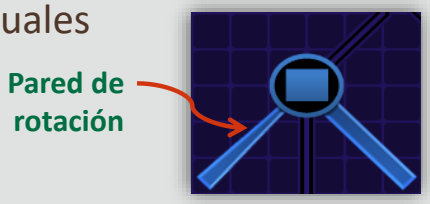
Ejercicio 1

- Juegue a los rompecabezas básicos del 8 al 11
  - https://objectstorage.uk-london-1.oraclecloud.com/n/lrvrlgaqj8dd/b/Games/o/JavaPuzzleBall/index.html
- Tenga en cuenta lo siguiente:
  - ¿Qué sucede al girar el objeto BlueWheel?
  - ¿De qué otra forma puede afectar a la rotación de los deflectores?



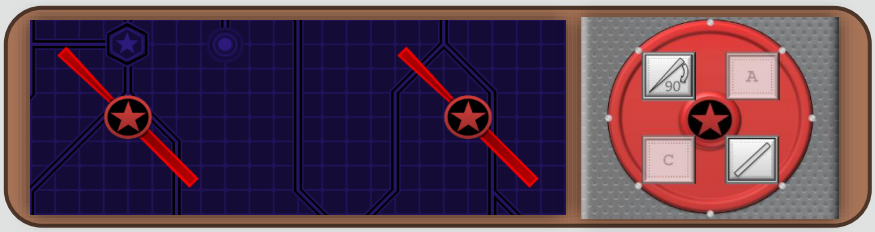
Análisis de Java Puzzle Ball

- ¿Qué sucede al girar el objeto BlueWheel?
  - Cambia la orientación de todos los objetos BlueBumper
  - Todos los objetos BlueBumper comparten la propiedad de orientación
  - La orientación puede estar representada por una variable estática
- ¿De qué otra forma puede afectar a la rotación de los deflectores?
  - Después de que la bola golpee una pared de rotación, cambia la rotación de los deflectores individuales
  - La rotación se puede representar mediante una variable de instancia



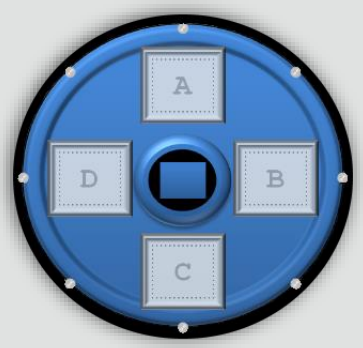
Variable estática: Orientación

- Todas las instancias comparten esta variable estática
- Las variables estáticas pertenecen a la clase, pero no a una instancia individual
- Por lo tanto, una variable estática se debe cambiar solo una vez por cada instancia que se vea afectada
- En el rompecabezas básico 11, al girar el objeto RedWheel se cambia la orientación de todos los objetos RedBumper



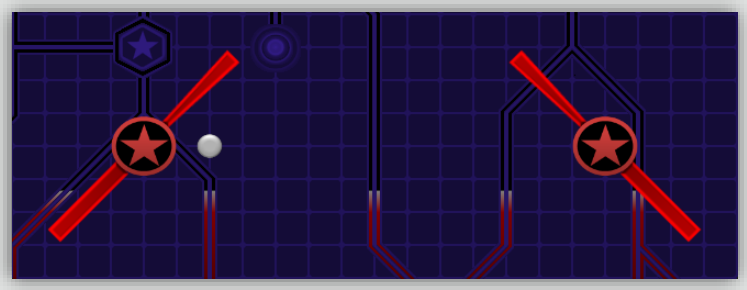
Variables estáticas sin instancias

- Se puede acceder a las variables estáticas, aunque no se hayan instanciado
- En el rompecabezas básico 11, el objeto BlueWheel se puede rotar para cambiar la propiedad orientation de todos los objetos BlueBumper
  - No hay ningún objeto BlueBumper para mostrar los efectos de este cambio



Variables de instancia: Rotación

- Existen variables de instancia únicas para cada instancia de un objeto
- Por lo tanto, las variables de instancia deben cambiarse para cada objeto individual
- En el rompecabezas básico 11, la rotación de un objeto RedBumper cambia después de que lo golpee la bola



¿Cuándo debe ser estático un campo?

- Algunos puntos que se deben tener en cuenta:
  - ¿El valor de este campo será distinto para cada objeto individual? ¿O será el mismo para todos los objetos?
  - ¿El campo describe la clase más que la describe cualquier objeto individual?
  - ¿Está repitiendo el mismo valor durante toda la clase?
  - ¿Es este valor una constante que se utilizará en los cálculos?
  - ¿Se tendrá que acceder a este valor antes de instanciar cualquier objeto?

Creación de variables estáticas

- Una variable se convierte en estática cuando en su declaración se incluye la palabra clave static
  - Inicialice las variables estáticas a medida que las declare
  - De lo contrario, las llamadas continuas al constructor podrían "inicializar" la misma variable estática varias veces

```
public class RedBumper{
   //Fields
   public static int orientation = 45;  //Static variable
   public int rotation;                 //Instance variable

   //Constructor
   public RedBumper(int rotation){
      this.rotation = rotation;
   }//end constructor RedBumper
}//end class RedBumper
```


Nota para los instructores: El cuadro de código de estas dos diapositivas debe estar en la misma posición.

Acceso a las variables estáticas en su clase

- Aunque las variables estáticas no se hayan inicializado en el constructor, se puede seguir accediendo a ellas
- Al igual que otras variables, se puede acceder a las variables estáticas desde su clase

```
public class RedBumper{
   //Fields
   public static int orientation = 45;  //Static variable
   public int rotation;                 //Instance variable
   …
   //Methods
   public void display(){
      System.out.println(orientation); //Access static var
      System.out.println(rotation);    //Access instance var
   }//end method display
}//end class RedBumper
```


Nota para los instructores: El cuadro de código de estas dos diapositivas debe estar en la misma posición.

Acceso a las variables estáticas desde cualquier parte

- Las variables estáticas pueden aparecer en constructores, en métodos o fuera de su clase
- La llamada a las variables estáticas fuera de su clase se basa en hacer referencia al nombre de la clase en lugar de a una variable de referencia específica

```
public class TestClass {
   public static void main(String[] args){
      int x;
      x = RedBumper.orientation;   //Access static variable

      RedBumper rb01 = new RedBumper(90); //Instance
      int y;
      y = rb01.rotation;          //Access instance variable
   }//end method main
}//end class TestClass
```

Ejercicio 2

- Continúe con la edición del proyecto PrisonTest
  - Se proporciona una versión de este programa en los archivos `PrisonTest_Student_7_6.java`, `Prisoner_Student_7_6.java` y `Cell_Student_7_6.java0`
- Modifique la clase Prisoner:
  - Incluir un campo prisonerCount entero estático
  - Este campo cuenta el número total de prisioneros instanciados
  - Inicializar este campo en 0
  - Incrementar este campo cada vez que se instancie un prisionero
  - Incluir el campo bookingNumber entero
  - Este campo se inicializa con el valor actual de prisonerCount
  - Imprimir bookingNumber y prisonerCount como parte del método display()
- Instancie algunos prisioneros y muestre su información

No necesita escribir métodos getter para este ejercicio

Introducción a los métodos estáticos

- Puede que lo haya advertido en el ejercicio anterior:
  - El método display() puede acceder a una variable estática
  - A las variables estáticas se puede acceder desde métodos no estáticos
- La mayoría de los métodos que ha escrito en este curso (excepto el método main) se consideran métodos de instancia
  - Los métodos de instancia no son estáticos
- Los métodos también se pueden convertir en estáticos

¿Cuándo debe ser estático un método?

- Algunos puntos que se deben tener en cuenta:
  - ¿El método estático leerá o modificará los campos?
  - ¿El método no leerá ni modificará los campos de un objeto concreto?
  - ¿Se deberá llamar al método antes de instanciar los objetos?
- Los métodos estáticos permiten manejar datos estáticos
  - A las variables estáticas se puede acceder desde métodos estáticos

Creación de métodos estáticos

- Un método se convierte en estático cuando en su declaración se incluye la palabra clave static

```
public class Prisoner{
   //Fields
   private static int prisonerCount = 0;  //Static variable
   private int bookingNumber;             //Instance variable

   //Methods
   public static void displayPrisonerCount(){ //Static method
      System.out.println(prisonerCount);
   }//end method displayPrisonerCount
}//end class Prisoner
```


Nota para los instructores: El cuadro de código de estas dos diapositivas debe estar en la misma posición.

Llamada a métodos estáticos en su clase

- Al igual que otros métodos, se puede llamar a los estáticos desde su clase
- Los métodos estáticos o de instancia pueden llamar a un método estático

```
public class Prisoner{
   private static int prisonerCount = 0;  //Static variable
   private int bookingNumber;             //Instance variable

   public static void displayPrisonerCount(){ //Static method
      System.out.println(prisonerCount);
   }//end method displayPrisonerCount
   public void callAnotherMethod(){         //Instance method
      displayPrisonerCount();
   }//end method callAnotherMethod
}//end class Prisoner
```


Nota para los instructores: El cuadro de código de estas dos diapositivas debe estar en la misma posición.

Llamada a métodos estáticos desde cualquier parte

- Se puede llamar a los métodos estáticos desde constructores, desde otros métodos o fuera de su clase
- La llamada a los métodos estáticos fuera de su clase se basa en hacer referencia al nombre de la clase en lugar de a una variable de referencia específica

```
public class TestClass {
   public static void main(String[] args){
      Prisoner.displayPrisonerCount();  //Call static method

      Cell cA1 = new Cell("A1", false, 1234);
      Prisoner bubba = new Prisoner("Bubba", 2.08, 4, cA1);
      bubba.display();                  //Call instance method
   }//end method main
}//end class TestClass
```

Ejercicio 3

- Continúe con la edición del proyecto `PrisonTest`
- Modifique la clase `Prisoner`:
  - Encapsule el campo prisonerCount Convierta este campo en privado y cree un método getter estático
  - Pruebe a convertir el método de visualización en estático
  - ¿Cuáles son las quejas de su IDE?
- En el método main:
  - Llame al método getter que acaba de crear e imprima el valor devuelto

¿Por qué se ha quejado su IDE?

- Los campos y métodos estáticos se pueden llamar sin instanciar un objeto
- Pero las variables de instancia deben estar asociadas a una instancia concreta
- Se produce una paradoja si un método estático intenta acceder a información sobre una instancia antes de que se cree
- Por lo tanto, Java no permite que los métodos estáticos contengan variables o métodos de instancia

```
public static void display(){
   System.out.println(prisonerCount);
   System.out.println(bookingNumber);
}//end method display
```

Escritura de campos static final

- Le animamos a que cree variables final de tipo static
  - Pero los motivos van más allá del ámbito de este curso
- Recuerde que los nombres de variables final...
  - Están en mayúsculas por convención
  - Utilizan guiones bajos (_) para separar las palabras

```
public class Prisoner{
   //Fields
    …
   private int bookingNumber;
   private static int prisonerCount = 0;
   public static final int MAX_PRISONER_COUNT = 100;
}//end class Prisoner
```


Las variables static son variables compartidas por todas las instancias de una clase.
Ejemplo: Una variable que representa un contador que aumenta de valor cada vez que se crea una instancia de la clase.

Las variables static final son variables compartidas por todas las instancias de la clase, pero el valor no puede cambiar.
Ejemplo: El tipo de interés o los tipos impositivos son fijos para todas las instancias de la clase. Además, se puede acceder a ellos sin crear una instancia de la clase y deben ser públicos.

Conversión de los campos primitivos static final en public

- La encapsulación evita que las variables se manipulen de un modo no deseado
- Pero no hay riesgo de que las primitivas public static final se modifiquen porque es imposible que cambien sus valores
- Esto resulta útil para constantes como π, e u otros valores usados constantemente en los cálculos
- Estas variables se llaman directamente en lugar de usar métodos getter

```
System.out.println(Math.PI);
System.out.println(Math.E);
```

Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Describir una variable estática y demostrar su aplicación dentro de un programa
  - Describir un método estático y mostrar su uso dentro de un programa
  - Comprender cómo se utiliza la palabra clave final con las variables estáticas


ORACLE
Academy