

# Variables y métodos estáticos

## Módulo 6

# Variables estáticas

Las variables estáticas son aquellas que se declaran utilizando la palabra reservada `static` antes de la definición de la variable.

Estas variables pertenecen a la clase, y por lo tanto, todos los Objetos pertenecientes a la clase la comparten y pueden acceder a ella.

Para acceder a una variable estática, utilizamos el nombre de la clase y no el objeto que la contiene.

Las variables estáticas también son conocidas como **class variables** (variables de clase).

# ¿Cuándo utilizar variables estáticas?

**#1: Para almacenar información que deban compartir TODOS los objetos de la clase.**

```
public class Student {  
    //Instance variable, every student has its own name  
    private String name;  
    //Instance variable, every student has a different studentID  
    private long studentID;  
    //Static variable, studentCount is shared between all instances  
    private static int studentCount;  
  
    public Student(String name, long studentID) {  
        this.name = name;  
        this.studentID = studentID;  
        Student.studentCount++;  
        System.out.println(this.name + " has enrolled. We are now " +  
                           Student.studentCount + " students in total!");  
    }  
}
```

# ¿Cuándo utilizar variables estáticas?

```
public class StudentDemo {  
    public static void main(String[] args) {  
        //Student.studentCount does not exist here  
        Student s1 = new Student("Juan", 2356687);    //studentCount = 1  
        Student s2 = new Student("Martha", 2356688); //studentCount = 2  
        Student s3 = new Student("Ana", 2356675);    //studentCount = 3  
    }  
}
```

## Output

Juan has enrolled. We are now 1 students in total!  
Martha has enrolled. We are now 2 students in total!  
Ana has enrolled. We are now 3 students in total!

# ¿Cuándo utilizar variables estáticas?

## #2: Para definir constantes

```
public class Circle {  
    //Instance variable, circle has an area  
    private double area;  
    //Static variable, constant  
    public static final double PI = 3.1416;  
  
    public Circle(double radius) {  
        this.area = Circle.PI * radius * radius;  
    }  
}
```

### Output

The value of PI is: 3.1416



La variable estática PI se puede acceder dentro o fuera de la clase utilizando: `Circle.PI`

```
public class CircleDemo {  
    public static void main(String[] args) {  
        System.out.println("The value of PI is: "+ Circle.PI);  
    }  
}
```

# ¿Cuándo utilizar cada tipo de variable?

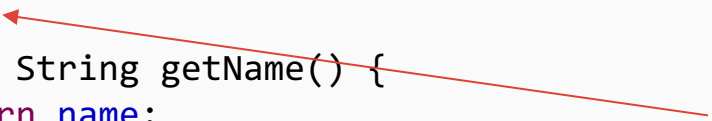
## Variables de instancia

- i. Información que pertenece a un solo objeto, y cambia de objeto en objeto.

**Dentro de la clase, se acceden utilizando el objeto `this`.**

```
public class Student {  
    //Instance variable, every student has its own name  
    private String name;  
    //Instance variable, every student has a different studentID  
    public long studentID;  
  
    public Student(String name, long studentID) {  
        this.name = name;  
        this.studentID = studentID;  
    }  
    public String getName() {  
        return name;  
    }  
}
```

**Dentro de la clase `Student`, podemos hacer referencia a cualquier variable de instancia utilizando el objeto `this`**



# ¿Cuándo utilizar cada tipo de variable?

## Variables de instancia

- i. Información que pertenece a un solo objeto, y cambia de objeto en objeto.


**Fuera de la clase, se acceden utilizando el objeto instanciado.**

```
public class StudentDemo {  
    public static void main(String[] args) {  
        Student s1 = new Student("Juan", 2356687);  
        Student s2 = new Student("Martha", 2356688);  
        System.out.println(s1.studentID);  
        System.out.println(s2.studentID);  
    }  
}
```

### Output

2356687


2356688



**Fuera de la clase Student, podemos hacer referencia a cualquier variable de instancia utilizando el nombre del objeto (s1 o s2).**

```
//static variable access  
Test.i1 = 10;
```

Acceder a variables  
estáticas, dentro o  
fuera de la clase



```
//instance variable access from outside the class  
//NOTE! this is a public variable  
Test t1 = new Test();  
t1.i2 = 5;
```

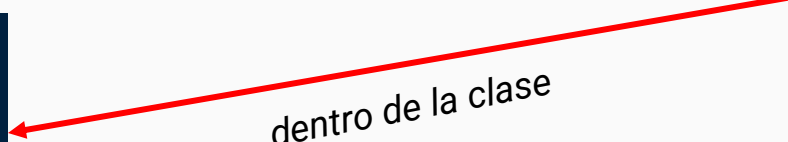
fuera de la clase



Acceder a  
variables de  
instancia

```
public void setI2(int i2){  
  
    //inside the class  
    this.i2 = i2;  
}
```

dentro de la clase





# Ejemplo

Una clase para  
representar a un alumno  
de preparatoria

## Instancia

- Nombre
- Apellido
- Altura
- Cursos inscritos del alumno
- Campus
- Fecha de Nacimiento
- Matrícula

## Estáticas

- Cantidad total de alumnos instanciados
- Promedio de altura de todos los alumnos
- Cantidad máxima de cursos que puede inscribir un alumno
- Cantidad mínima de cursos que puede inscribir un alumno

# Variables estáticas y variables de instancia



## Variables de Instancia

**Nombre:** Jose Ramirez  
**Matrícula:** A04458875  
**Fecha de nac:** 10 octubre 1990  
**Cursos inscritos:** 5



**Nombre:** María Meza  
**Matrícula:** A017895563  
**Fecha de nac:** 11 junio 1995  
**Cursos inscritos:** 7



**Nombre:** Pamela Alavés  
**Matrícula:** A099877456  
**Fecha de nac:** 12 marzo 2000  
**Cursos inscritos:** 6

## Variables estáticas

**Cantidad de alumnos:** 3  
**Promedio cursos inscritos:** 6  
**Máximos cursos inscritos permitidos:** 7

# Métodos estáticos

Los métodos estáticos son aquellos que se declaran utilizando la palabra reservada **static** previo a la definición del método.

Los métodos estáticos pertenecen a la Clase, y no pueden depender o estar atados a ningún Objeto. Para funcionar sólo pueden hacer uso de los parámetros de entrada, o de otras variables estáticas en la clase.

```
public class Test {  
    //static variable  
    private static int i1;  
  
    //static method  
    public static int multiply(int a, int b) {  
        return a*b;  
    }  
  
    //static method  
    public static void multiplyByI1(int a) {  
        return i1*a;  
    }  
}
```

# Métodos estáticos

1. Utilizan sólo los parámetros de entrada y otros métodos estáticos o variables estáticas de la clase.
2. Métodos de tipo *utility* (conversión de datos, manipulación de Strings, ordenamiento de arreglos) que no debe cambiar.
3. Métodos setter / getter para variables estáticas
4. Si hay código común (validaciones, cálculos) que puedan ser compartidos, puede abstraerse a un método estático.
5. Generalmente tienen mejor desempeño, pues evitas la instanciación y memoria del objeto.
6. No requieren que exista una instancia de un objeto para poder ser invocados.
7. Dentro o fuera de la clase, se acceden utilizando el nombre de la clase.

# Métodos de instancia

1. Pueden modificar o leer las variables de instancia de la clase.
2. Requieren forzosamente que se instancie un objeto previo a su invocación.
3. Métodos setter / getter para variables de instancia
4. Sin un objeto, llamar este método no tiene sentido.
5. Dentro de la clase, se invocan utilizando el objeto ***this***.
6. Fuera de la clase, se invocan utilizando el nombre objeto instanciado.

# Métodos estáticos y métodos de instancia

## Método estático

Nota: Observar que el método abs se accede directamente con el nombre de la clase Math.

```
int x = Math.abs(-10);
```

## Método de instancia

Nota: Observar que es necesario instanciar el objeto s1 de la clase Scanner.

```
Scanner s1 = new Scanner(System.in);  
int y = s1.getNext();
```

# Una clase para representar a un alumno de la Prepa Tec



## Instancia

1. Actualizar el nombre del alumno
2. Actualizar el apellido del alumno
3. Actualizar la altura del alumno
4. Imprimir la información del alumno
5. Getter para la variable cursos inscritos del alumno
6. Actualizar el Campus
7. Leer la fecha de nacimiento
8. Imprimir en consola la matrícula

# Una clase para representar a un alumno de la Prepa Tec



## Estáticos

1. Leer la cantidad total de alumnos instanciados (variable estática).
2. Calcular el promedio de altura de todos los alumnos a partir de un arreglo
3. Leer la cantidad máxima de cursos que puede inscribir un alumno (constante)
4. Leer la cantidad mínima de cursos que puede inscribir un alumno (constante)
5. Ordenar una lista recibida de alumnos en orden alfabético.



# Caso de Estudio: Clase DollarFormat



# Dollar Format

Si manejas variables de tipo `double` para almacenar dinero, los programas que utilices deben imprimir los montos con un formato adecuado.

```
Your cost, including tax is $3.6666666666666665
```

Sería preferible que se impriman de la siguiente forma:

```
Your cost, including tax is $3.67
```

Diseñemos una clase **DollarFormat** con dos métodos estáticos **write** y **writeln** que nos ayuden a producir las cantidades correctamente formateadas. De tal forma que el siguiente código produzca el resultado esperado:

```
System.out.println("Your cost, including tax is ");  
DollarFormat.writeln(amount);
```

# Método `write`

1. Redondear los centavos a dos decimales.
2. Agregar el signo de dólares \$
3. Agregar salto de línea con el método `writeln`.

## ¿Qué tipo de método usar?

Como es un método que sólo dependerá de los parámetros de entrada, puede ser un método estático

Recibir una variable de tipo `double`, e imprimir en pantalla la cantidad formateada con el formato:

- \$ {Dolares}.{centavos}

Redondear la cantidad recibida a dos decimales

Casos de prueba:

1. 12.78
2. 100.00
3. 10.00
4. 1.00
5. 0.10
6. 0.01
7. 0.001
8. 0.00

# Método write

Algoritmo:

1. Calcular la parte entera del número
2. Calcular los centavos.
3. Imprimir:
  - Símbolo de dólares
  - Dólares enteros
  - Punto
  - Si la cantidad de centavos  $< 10$ , imprimir 0
  - Cantidad centavos

# Método write

```
public static void write(double amount) {  
    int cents = (int)(Math.round(amount * 100));  
    cents = cents % 100;  
    int dollars = (int)amount;  
  
    System.out.print("$");  
    System.out.print(dollars);  
    System.out.print(".");  
    if (cents < 10) {  
        System.out.print("0");  
    }  
    System.out.print(cents);  
}
```

# Método `writeln`

- Reutilizar el método `write`
- Agregar un salto de línea

## ¿Qué tipo de método usar?

Como es un método que sólo dependerá de los parámetros de entrada, puede ser un método estático

```
public static void writeln(double amount) {  
    write(amount);  
    System.out.println();  
}
```

# writeln

```
public static void main(String[] args) {  
    DollarFormat.writeln(100.00);  
    DollarFormat.writeln(10.00);  
    DollarFormat.writeln(1.00);  
    DollarFormat.writeln(0.10);  
    DollarFormat.writeln(0.01);  
    DollarFormat.writeln(0.001);  
    DollarFormat.writeln(0.00);  
    DollarFormat.writeln(12.78);  
    DollarFormat.writeln(11.456);  
}
```

## Output

```
$100.00  
$10.00  
$1.00  
$0.10  
$0.01  
$0.00  
$0.00  
$12.78  
$11.46
```

# Y los números negativos?



```
public static void main(String[] args) {  
    DollarFormat.writeln(-100.00);  
    DollarFormat.writeln(-10.00);  
    DollarFormat.writeln(-1.00);  
    DollarFormat.writeln(-0.10);  
    DollarFormat.writeln(-0.01);  
    DollarFormat.writeln(-0.001);  
    DollarFormat.writeln(-0.00);  
    DollarFormat.writeln(-12.78);  
    DollarFormat.writeln(-11.456);  
}
```

```
$-100.00  
$-10.00  
$-1.00  
$0.0-10  
$0.0-1  
$0.00  
$0.00  
$-12.0-78  
$-11.0-46
```



# Cambiar los métodos

Crear nuevo método:

```
void writePositive(double amount)
```

Modificar el método:

```
void write(double amount)
```

```
public static void write(double amount) {  
    if (amount < 0) {  
        System.out.print("-");  
    }  
    DollarFormat.writePositive(Math.abs(amount));  
}
```

```
private static void writeLn(double amount) {  
    write(amount);  
    System.out.println();  
}
```

```
private static void writePositive(double amount) {  
    int cents = (int)(Math.round(amount * 100));  
    cents = cents % 100;  
    int dollars = (int)amount;  
  
    System.out.print("$");  
    System.out.print(dollars);  
    System.out.print(".");  
    if (cents < 10) {  
        System.out.print("0");  
    }  
    System.out.print(cents);  
}
```

```
public static void main(String[] args) {  
    DollarFormat.writeln(100.00);  
    DollarFormat.writeln(10.00);  
    DollarFormat.writeln(1.00);  
    DollarFormat.writeln(0.10);  
    DollarFormat.writeln(0.01);  
    DollarFormat.writeln(0.001);  
    DollarFormat.writeln(0.00);  
    DollarFormat.writeln(12.78);  
    DollarFormat.writeln(11.456);  
  
    DollarFormat.writeln(-100.00);  
    DollarFormat.writeln(-10.00);  
    DollarFormat.writeln(-1.00);  
    DollarFormat.writeln(-0.10);  
    DollarFormat.writeln(-0.01);  
    DollarFormat.writeln(-0.001);  
    DollarFormat.writeln(-0.00);  
    DollarFormat.writeln(-12.78);  
    DollarFormat.writeln(-11.456);  
}
```

```
$100.00  
$10.00  
$1.00  
$0.10  
$0.01  
$0.00  
$0.00  
$12.78  
$11.46  
-$100.00  
-$10.00  
-$1.00  
-$0.10  
-$0.01  
-$0.00  
$0.00  
-$12.78  
-$11.46
```