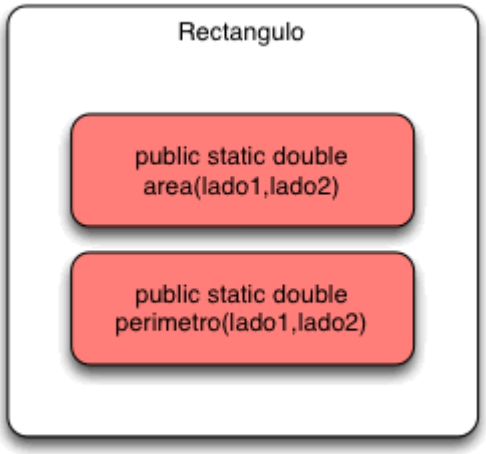
Hace poco alguién me ha preguntado a traves del blog para que sirve un Java static method y que diferencia hay con los métodos de instancia (instance methods) .La consulta tiene como punto de partida otros de los articulos el de el concepto de interface .Me ha parecido interesante hablar un poco de este tema porque sobre todo a la gente que comienza le genera muchas dudas. Un método estático es un método que tiene sentido invocarla sin crear previamente ningun objeto .

Vamos a ver un ejemplo sencillo supongamos que necesitamos crear la clase Rectangulo que calcule el area y el perímetro . Mucha gente que comienza diseñaría una clase como la siguiente .



Vamos a ver su código fuente para aclararnos del todo :

```
package com.arquitecturajava.estatico;

public class Rectangulo {
 public static double area(int lado1,int lado2) {

 return lado1*lado2;
 }
```

```
 public static double perimetro(int lado1,int lado2) {

 return lado1*2+lado2*2;
 }
}
```

Una vez que hemos creado la clase la utilizamos en un programa de consola.

```
package com.arquitecturajava.estatico;

public class Principal {

public static void main(String[] args) {
 imprimirArea(2,2);

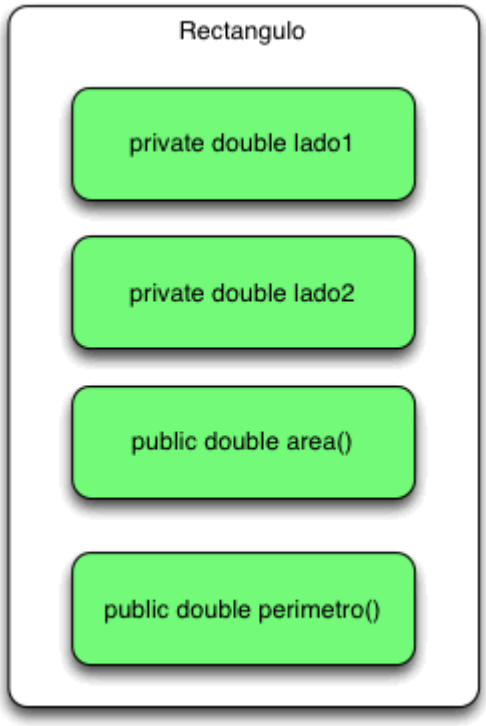
 }

public static void imprimirArea(int lado1,int lado2) {
 System.out.println(Rectangulo.area(lado1, lado2));
 }
}
```

Orientación a Objeto y metodos estáticos

Todo nos parece bastante correcto y estamos contentos con la solución .Hemos usado métodos un Java static method que nos permiten invocar funcionalidad sin crear objetos.Ahora bien puede que otra persona enfoque de una forma mas orientada a objeto y

construya una clase como la siguiente.



Vamos a ver su código fuente :

```
public class Rectangulo {

    private int lado1;
    private int lado2;

    public Rectangulo(int lado1, int lado2) {
        super();
        this.lado1 = lado1;
        this.lado2 = lado2;
    }

    public double area() {
```

```
        return lado1 * lado2;
    }

    public double perimetro() {
        return (lado1 * 2) + (lado2 * 2);
    }
}
```

Si creamos una programa de consola será bastante parecido al anterior

```
package com.arquitecturajava.instancia;

import java.util.ArrayList;

public class Principal {

    public static void main(String[] args) {
        imprimirArea(new Rectangulo(2, 2));
    }

    public static void imprimirArea(Rectangulo r) {
        System.out.println(r.area());
    }
}
```

¿Cual les mejor?

¿Con cual de los dos enfoques nos quedamos? .A veces es dificil de ver y las discusiones entre los desarrolladores son frecuentes. Supongamos que queremos calcular el area de varios elementos a la vez .¿Como quedarían los programas?. Vamos a ver en primer lugar el de los métodos estáticos.

```
public static void main(String[] args) {
    int[][] datos = new int[2][2];

    datos[0][0] = 1;
    datos[0][1] = 1;
    datos[1][0] = 2;
    datos[1][1] = 2;
    imprimirVarios(datos);
}

public static void imprimirArea(int lado1, int lado2) {
    System.out.println(Rectangulo.area(lado1, lado2));
}

public static void imprimirVarios(int[][] datos) {
    for (int i = 0; i < datos.length; i++) {
        System.out.println(Rectangulo.area(datos[i][0], datos[i][1]));
    }
}
```

A continuación mostramos el de programación orientada a objeto .

```
package com.arquitecturajava.instancia;

import java.util.ArrayList;

public class Principal {

    public static void main(String[] args) {
        ArrayList<Rectangulo> listaRectangulos = new
ArrayList<Rectangulo>();
```

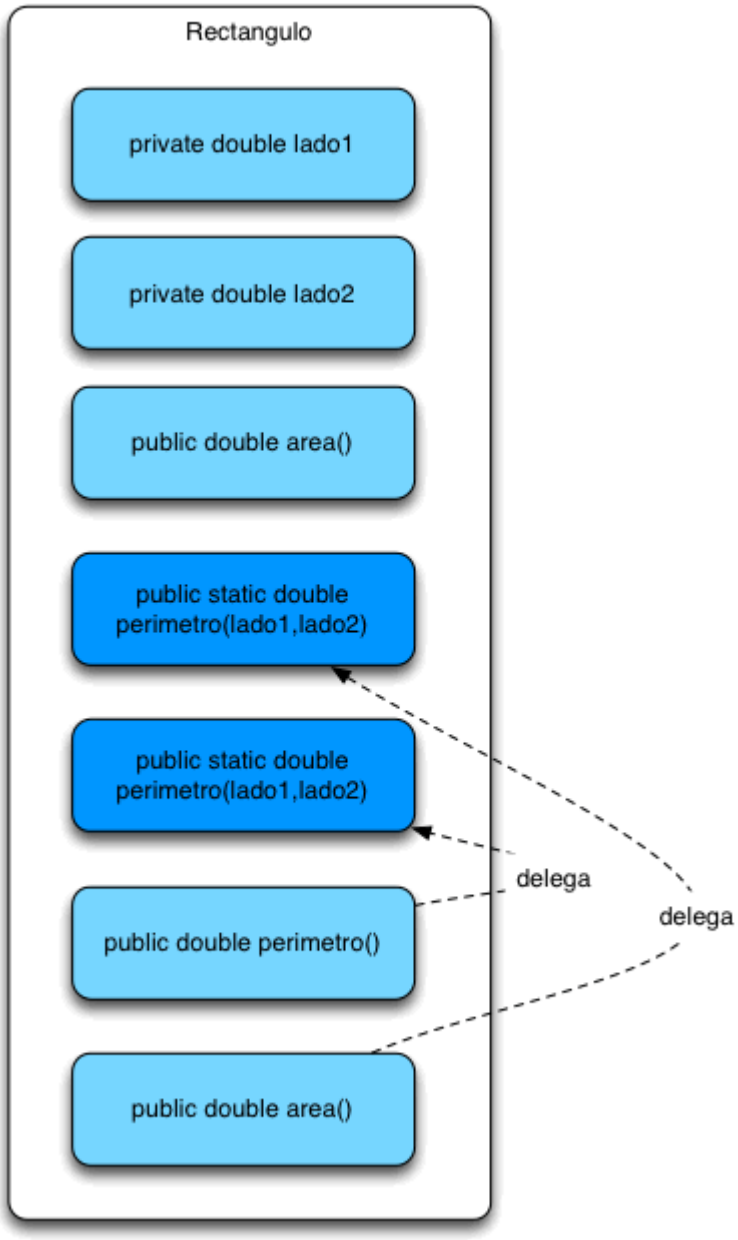
```
        listaRectangulos.add(new Rectangulo(1, 1));
        listaRectangulos.add(new Rectangulo(2, 2));
        imprimirArea(new Rectangulo(2, 2));
    }

    public static void imprimirArea(Rectangulo r) {
        System.out.println(r.area());
    }

    public static void imprimirVarios(ArrayList<Rectangulo> lista) {
        for (Rectangulo r : lista) {
            System.out.println(r.area());
        }
    }
}
```

Java Static method y diseño

Nos podemos dar cuenta que la segunda versión es mucho mas sencilla que la primera de entender que la primera esto es algo que la programación orientada a objeto facilita. Aún así habrá gente que siga considerando que los métodos estáticos pueden sernos utiles en algún momento . Bueno si ese es el caso podemos rediseñar la clase de la siguiente forma y tendremos las ventajas de la programación orientada objeto y las ventajas de la programación mas estructurada con métodos estáticos.



Vamos a ver el código final :

```
package com.arquitecturajava.diseño;

public class Rectangulo {
```

```
private int lado1;
private int lado2;

public Rectangulo(int lado1, int lado2) {
    super();
    this.lado1 = lado1;
    this.lado2 = lado2;
}

public int getLado1() {
    return lado1;
}

public void setLado1(int lado1) {
    this.lado1 = lado1;
}

public int getLado2() {
    return lado2;
}

public void setLado2(int lado2) {
    this.lado2 = lado2;
}

public double area() {
    return Rectangulo.area(getLado1(), getLado2());
}

public double perimetro() {
    return Rectangulo.perimetro(getLado1(), getLado2());
```

```
    }

    public static double area(int lado1, int lado2) {
        return lado1 * lado2;
    }

    public static double perimetro(int lado1, int lado2) {
        return lado1 * 2 + lado2 * 2;
    }
}
```

Conclusiones

Espero que el ejemplo ayude aclarar un poco las dudas sobre Java Static method y como podemos mezclar unos métodos y otros contruyendo una solución más elegante.

- Default methods flexibilidad vs solidez
- Java Supplier Interface y Factories
- Cursos Arquitectura JAVA Fundae
- Java interface private method y como usarlo