Hacerse la pregunta sobre Java Object Size o cuanto ocupa un objeto en Java en memoria es bastante común. En muchas ocasiones nos encontramos desarrollando aplicaciones que instancia muchos objetos. En este tipo de situaciones no es infrecuente tener problemas de gestión de memoria.La primera pregunta que nos viene a la cabeza es ¿Cuanto ocupan en memoria los objetos que acabamos de construir ? . Vamos a abordar este problema de una forma sencilla. El primer paso es instalarlos la siguiente librería.

```
<dependency>
    <groupId>com.github.jbellis</groupId>
    <artifactId>jamm</artifactId>
    <version>0.3.1</version>
</dependency>
```

Esta pequeña librería nos aporta una clase que nos permite saber lo que ocupa cada objeto en memoria. Vamos a ver un par de ejemplos de su uso.

```
package com.arquitecturajava;

import org.github.jamm.MemoryMeter;


public class Principal {

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                MemoryMeter memoria= new MemoryMeter();
                System.out.println(memoria.measureDeep(new
```

```
byte[1000]));
                System.out.println(memoria.measureDeep(new
byte[2000]));
        }

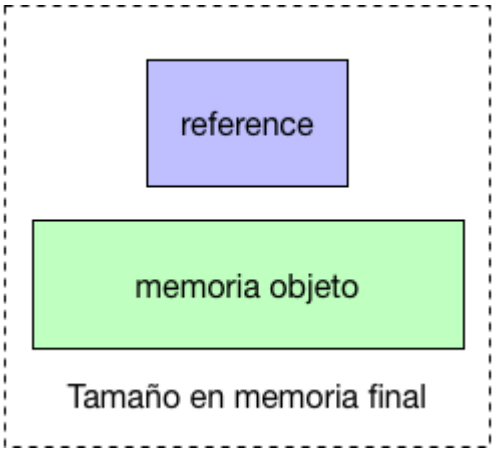
}
```

La clase es muy sencilla de utilizar , dispone de un método measureDeep que nos indica cuanto ocupa un objeto o array determinado. En este caso hemos construido dos arrays uno de 1000 y el otro de 2000. Vamos a ver que resultado se imprime por pantalla.



Java Object Size JVM Overhead

Debería haber imprimido 1000 y 2000 bytes pero ha imprimido un valor un poco superior esto es debido a que un array en Java no solo ocupa su espacio sino que hay que añadir un espacio adicional para su referencia etc. Esto es lo que comunmente se denomina JVM Overhead



Una vez tenemos esto podemos usar esta clase para calcular el espacio en memoria de cualquier estructura. Por ejemplo supongamos que disponemos del concepto de Compra con una serie de productos.

```
package com.arquitecturajava;

public class Producto {
	private int numero;
	private String concepto;
	private int importe;
	private String categoria;
	private String descripcionCategoria;
	public int getNumero() {
		return numero;
	}
	public void setNumero(int numero) {
		this.numero = numero;
	}
	public String getConcepto() {
		return concepto;
	}
	public void setConcepto(String concepto) {
		this.concepto = concepto;
	}
	public int getImporte() {
		return importe;
	}
	public void setImporte(int importe) {
		this.importe = importe;
```

```
	}
	public String getCategoria() {
		return categoria;
	}
	public void setCategoria(String categoria) {
		this.categoria = categoria;
	}
	public String getDescripcionCategoria() {
		return descripcionCategoria;
	}
	public void setDescripcionCategoria(String
descripcionCategoria) {
		this.descripcionCategoria = descripcionCategoria;
	}
	public Producto(int numero, String concepto, int importe,
String categoria, String descripcionCategoria) {
		super();
		this.numero = numero;
		this.concepto = concepto;
		this.importe = importe;
		this.categoria = categoria;
		this.descripcionCategoria = descripcionCategoria;
	}

}
```

Acabamos de definir la clase Producto ,es momento de diseñar un programa que genere una lista de productos y calculemos el espacio que ocupa en memoria:

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;

import org.github.jamm.MemoryMeter;


public class Principal2 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		List<Producto> productos = new ArrayList<Producto>();

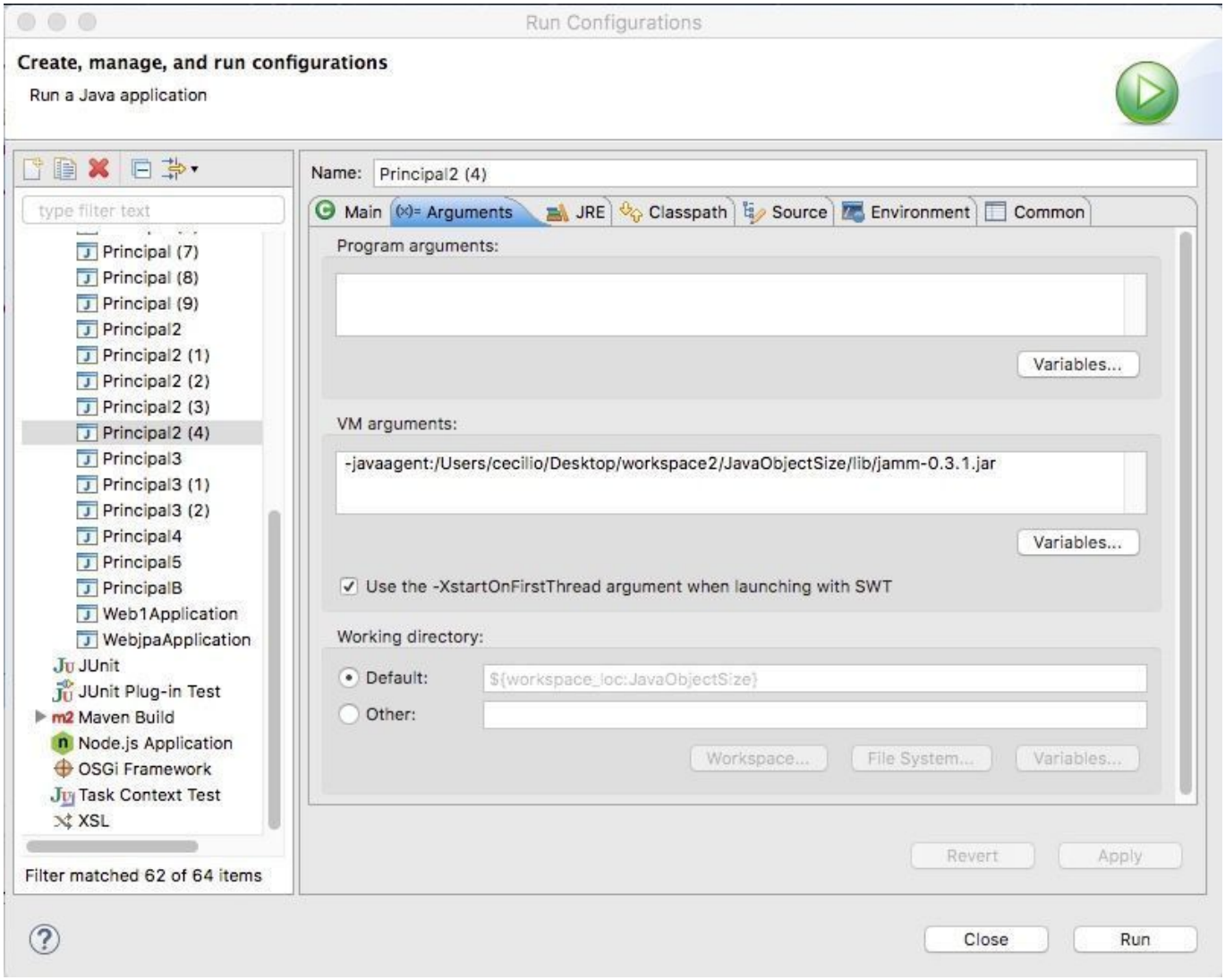
		for (int i = 0; i < 20; i++) {

			Producto c = new Producto(1, "ordenador", 500,
"categoria1", "texto");
			productos.add(c);

		}
		MemoryMeter memoria= new MemoryMeter();
		System.out.println(memoria.measureDeep(productos));
	}

}
```

Estos nos imprimirá el Java Object Size de nuestra lista de productos. Para que el código funcione correctamente tendremos que añadir un parámetro de arranque a la máquina virtual definiendo un javagent:



Una vez hecho esto , ejecutamos y el resultado se imprime en pantalla:



La lista de productos que hemos construido ocupa prácticamente 1k en memoria. Esta información nos puede ser útil cuando valoremos en nuestras aplicaciones el consumo de

memoria . Por ejemplo supongamos que almacenamos este concepto de lista de productos en la sesión de la aplicación web. Cada usuario consumirá entre 1 y 3 kilobytes de memoria. Si tenemos en un momento determinado 2000 sesiones activamos podemos calcular que el consumo será como máximo 6 megas.

Otros artículos relacionados:

1. Java HttpSession Timeout
2. String ,StringBuffer y rendimiento
3. Usando Java Session en aplicaciones web