

## CUESTIONES Y EJERCICIOS

**ORGANIZACIÓN DEL CÓDIGO FUENTE POR PROYECTOS**


Para tener organizado el código de los ejercicios y poder localizarlos rápidamente se recomienda llamar a los proyectos de NetBeans del siguiente modo: “Ux.Ez” donde x será el número de la unidad didáctica, z será el número del ejercicio.

Por ejemplo: **U1.E3** para el ejercicio 3 de la unidad didáctica 1

Opcionalmente podemos añadir al final un punto y una palabra que describa algún elemento del ejercicio para que sea más fácil localizarlo.

Ejemplo: **U1.E1.Abeja**

1. Comienza un proyecto nuevo en NetBeans llamado “U1.E1.Abeja”. A continuación, crea una nueva clase Abeja. Ve a Moodle y copia el código de la clase Abeja que encontrarás colgado. A continuación, debes crear una clase PruebaAbeja2 que, teniendo en cuenta la imagen que se muestra en la siguiente página, realice las siguientes operaciones:
  1. Defina y cree un objeto de la clase Abeja
  2. A continuación, la abeja debe volar, desplazarse a la posición (2,0), posarse y cambiar el número de cargas de néctar a 1.
  3. La abeja debe repetir la operación anterior pero ahora irá a la posición (2,2) y cambiará el número de cargas de néctar a 2.
  4. La abeja repite la operación anterior pero ahora pasará por la flor de la posición (0,2) y cambiará el número de cargas de néctar a 3.
  5. Por último, la abeja debe imprimir el valor de sus propiedades haciendo uso de los métodos imprimeXXX()

		Eje X		
		0	1	2
Eje Y	0			
	1			
	2			

2. Se desea comenzar un nuevo proyecto llamado U1.E2.Bombilla que codifique la clase *Bombilla* siguiendo las siguientes especificaciones:

Bombilla
+ marca: texto + potencia: entero + encendida: lógico
+ cambiaMarca (nuevaMarca: texto) + imprimeMarca () + cambiaPotencia (nuevaPotencia: entero) + imprimePotencia () + encender () + apagar () + imprimeEstado ()

El método *imprimeEstado()* debe mostrar el texto: "Encendida?: true" o "Encendida?: false" según proceda.

Una vez codificada, se debe salvar como *Bombilla.java* y compilarse depurando los errores que se detecten.

Por último, se debe crear un nuevo fichero llamado *PruebaBombilla.java* que contenga una clase *PruebaBombilla* que tendrá un método *main* y que realizará las siguientes operaciones:

- Definir y crear un objeto de la clase Bombilla
- Imprimir su marca y su potencia
- Cambiar la marca a "Phillips"
- Cambiar la potencia a 100
- Encender, apagar y encender de nuevo.
- Imprimir la marca, potencia y estado.

3. Comienza un proyecto nuevo en NetBeans llamado “U1.E3.Triangulo”. Ahora queremos codificar una clase que represente un triángulo. Para ello, la vamos a modelar así:

Triangulo
+ tamLado1: número real + tamLado2: número real + tamLado3: número real
+ cambiaLado1 (nuevoTamanio: real) + cambiaLado2 (nuevoTamanio: real) + cambiaLado3 (nuevoTamanio: real) + imprimePerimetro () + imprimeDescripcion ()

Donde:

- `imprimePerimetro` debe mostrar el texto: “El perímetro del triángulo es: xxxxx”
- `imprimeDescripcion` debe mostrar el texto: “Soy un triángulo y el tamaño de mis lados es lado1=xxxx, lado2=www y lado3=zzzz” siendo xxxx, www y zzzz el valor de las respectivas propiedades.

Una vez codificada la clase `Triangulo`, se debe salvar como *Triangulo.java* y compilarse depurando los errores que se detecten.

Ahora, debes crear otra clase llamada `PruebaTriangulo` en otro fichero llamado `PruebaTriangulo.java`. Esta clase tendrá un método *main* en el que:

- Debes crear un objeto de la clase Triángulo y asignarlo a una referencia llamada *trian*.
- Debes cambiar el tamaño de los lados a los siguientes valores: 23.32, 12.21, 44.34.
- Por último, llama a los métodos *imprimePerimetro* e *imprimeDescripcion* y comprueba si los textos se imprimen correctamente.

4. Crea un nuevo proyecto y codificar la clase *Persona* a partir de su diagrama UML:

Persona
+ nombre: texto + edad: entero + trabaja: lógico + casada: lógico
+ cambiaNombre(nuevoNombre:texto) + imprimeNombre () + cambiaEdad (nuevaEdad: entero) + imprimeEdad () + consigueTrabajo () + pierdeTrabajo () + imprimeEstadoTrabajo () + seCasa () + seDivorcia () + imprimeEstadoCivil ()

El método *imprimeEstadoTrabajo()* debe mostrar el texto: "Trabaja?: true" o "Trabaja?: false" según proceda.

Una vez codificada, se debe salvar como *Persona.java* y compilarse depurando los errores que se detecten.

Por último, se debe crear un nuevo fichero llamado *PruebaPersona.java* que contenga una clase *PruebaPersona* que tendrá un método *main* y que realizará las siguientes operaciones en orden:

- Definir y crear un objeto 'p1' de la clase Persona
- Definir y crear un objeto 'p2' de la clase Persona
- Cambiar el nombre y la edad de 'p1' poniendo los valores "Pepe" y 18
- Cambiar el nombre y la edad de 'p2' poniendo los valores "Pepa" y 19
- Imprimir el nombre, la edad, el estado laboral y el estado civil de "Pepe"
- Imprimir el nombre, la edad, el estado laboral y el estado civil de "Pepa"
- Ahora "Pepe" consigue un trabajo, después lo pierde y consigue otro trabajo de nuevo.
- "Pepa" consigue un trabajo.
- Cambiamos la edad de "Pepe" y "Pepa", ahora tienen 30 y 31
- "Pepe" se casa y "Pepa" también
- "Pepe" se divorcia, pero "Pepa" no.
- Imprimir el nombre, la edad, el estado laboral y el estado civil de "Pepe"
- Imprimir el nombre, la edad, el estado laboral y el estado civil de "Pepa"

5. Crear un nuevo proyecto y codifica la clase Vehículo con el siguiente diagrama UML:

Vehículo
+ numRuedas: entero + potencia: real + litrosEnDeposito: real + consumoPorKm: real + arrancado: lógico
+ setNumRuedas(nuevoNumRuedas: entero) + setPotencia (nuevaPotencia: real) + setConsumoPorKm (nuevoConsumoPorKm: real) + reponerCombustible(numLitrosRepuesto: real) + recorrerDistancia (numKm: real) + arrancar() + apagar() + imprimeNumRuedas() + imprimePotencia() + imprimeAutonomiaEnKm() + imprimeLitrosEnDeposito()

Donde:

- Los métodos *setXXX* establecen un nuevo valor en la propiedad correspondiente.
- Los métodos *arrancar* y *apagar* modifican el valor de la propiedad *arrancado*.
- El método *reponerCombustible* añade una cantidad de litros a los ya existentes en el depósito.
- El método *recorrerDistancia* modificar el valor de la propiedad *litrosEnDeposito* actualizándolo de modo que se aplique el *consumoPorKm* tantas veces como kilómetros se hayan recorrido.
- Los métodos *imprimeXXXX* imprimen un texto en la pantalla en el que se muestre el valor de la propiedad correspondiente.
- El método *imprimeAutonomiaEnKm()* debe imprimir en pantalla el texto: "El vehículo consume XXX litros/km, tiene YYY litros en el depósito, así que puede recorrer ZZZ km". Debes usar una variable para calcular el número de km de autonomía (ZZZ).

Por último, realiza una clase *PruebaVehiculo* con un método *main* que permita probar un objeto de la clase Vehículo y que pruebe todos los métodos de esta clase.

6. Retoca una clase Vehiculo del ejercicio *extra* anterior para que incorpore la nueva información que está en cursiva.

Vehiculo
... + <i>kmRecorridos: real</i>
... + <i>imprimeKmRecorridos()</i>

Además, debes tener en cuenta las siguientes consideraciones:

- *imprimeKmRecorridos* debe imprimir el texto “Kilómetros recorridos: XX”, donde XX se reemplazará por el valor de la propiedad *kmRecorridos*.
- Vamos a retocar el método *recorrerDistancia* para que además de actualizar el número de litros que queda en el depósito, actualice también la propiedad *kmRecorridos*.

A continuación, crea una clase *PruebaVehiculo2* que cree dos objetos Vehículo v1 y v2 y que realice las siguientes acciones:

- v1 será un vehículo de 4 ruedas con 100 caballos de vapor, con un consumo por kilómetro de 0.06 litros. Después repone 30 litros de combustible, arranca, recorre 100 km, para, arranca y recorre otros 200 km más. Después usa todas las propiedades *imprimeXXXX* para imprimir su estado.
- v2 será un vehículo de 2 ruedas con 80 caballos de vapor, con un consumo por kilómetro de 0.02 litros. Después repone 20 litros de combustible, arranca, recorre 220 km, para, arranca y recorre otros 120 km más. Después usa todas las propiedades *imprimeXXXX* para imprimir su estado.

Comprueba si los resultados que ofrece tu programa son correctos.

Por último, usa el documento “Plantilla de memoria.docx” y rellénalo con las referencias v1 y v2 y los dos objetos en su estado al finalizar el programa. Invéntate las direcciones de memoria en las que se alojan los distintos elementos.

7. Partiendo de la clase *Bombilla* del ejercicio 2 vamos a modificarla y obtener otra clase llamada *BombillaInteligente* que añadirá lo siguiente:

1. Una nueva propiedad llamada *numVecesEncendida* de tipo entero que será incrementada en una unidad cada vez que se llame al método *encender()*.
2. Un nuevo método con el siguiente prototipo: *public int obtieneNumVecesEncendida()* que será capaz de devolver el número de veces que ha sido encendida la bombilla.

También se debe crear una clase *PruebaBombillaInteligente* que cree un objeto de la clase *BombillaInteligente* y haga uso del nuevo método.

8. Partiendo de la clase anterior vamos a crear una nueva clase llamada *Prueba2Bombillas* que creará **dos objetos** de la clase *BombillaInteligente*. La primera bombilla se encenderá y apagará 3 veces y la segunda bombilla solo una vez. Por último, se deben imprimir el número de veces que se ha impreso cada bombilla.
9. Escribir la clase *Circunferencia* y probarla con una clase *PruebaCircunferencia* siguiendo el siguiente diagrama:

Circunferencia
+ radio: real
+ estableceRadio (nuevoRadio: real)
+ calculaPerimetro() <b>devuelve:</b> real
+ calculaSuperficie() <b>devuelve:</b> real

10. Retomar la clase *Persona* del ejercicio 4 y complementarla con:

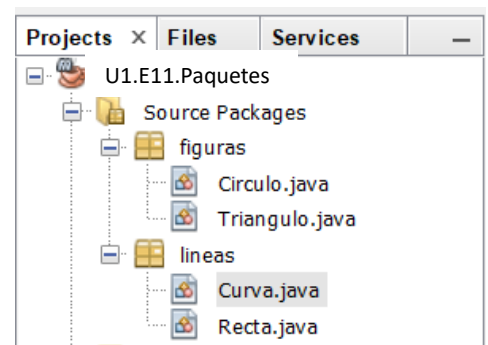
- Un método: **public void cumpleAnios()** que aumenta en 1 la propiedad *edad* de la persona.
- Un método como sigue: **public int obtieneNumAnios()** que devuelve el valor de la propiedad *edad*.

Por último, se debe crear una clase *PruebaCumplePersona* que tenga un método *main* que cree dos objetos *p1* y *p2* de la clase *Persona*. Y los trataremos del siguiente modo:

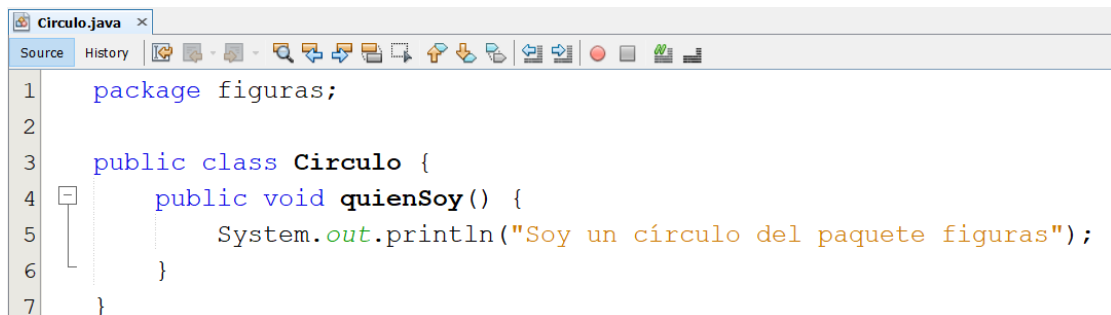
- Cambiaremos el nombre de *p1* por "Carmen" y su edad por 15.
- Cambiaremos el nombre de *p2* por "Alberto" y su edad por 16.
- Imprimir el nombre, la edad, el estado laboral y el estado civil de "Carmen"
- Imprimir el nombre, la edad, el estado laboral y el estado civil de "Alberto"
- A continuación haremos que *p1* cumpla 4 años utilizando el método *cumpleAnios()*.
- A continuación haremos que *p2* cumpla 3 años utilizando el método *cumpleAnios()*.
- Por último, tenemos que imprimir la edad de ambos objetos haciendo uso del método *obtieneNumAnios()*

11. Abre un nuevo proyecto llamado *U1.E11.Paquetes* y crea la siguiente estructura de clases y paquetes. Las clases deben estar vacías, no tendrán ni propiedades ni métodos.

A continuación, abre un explorador de carpetas y busca la estructura de carpetas y archivos que se ha creado.



12. Sin cerrar el proyecto anterior, abre un nuevo proyecto llamado *U1.E12.ImportaPaquetes*. Ahora usa Ctrl-C y Ctrl-V para pegar en este proyecto los paquetes *figuras* y *líneas* del proyecto anterior. A continuación, modifica el código de las 4 clases de modo que contengan un único método como el que sigue:



```
1 package figuras;
2
3 public class Circulo {
4     public void quienSoy() {
5         System.out.println("Soy un círculo del paquete figuras");
6     }
7 }
```

Por último, crea una nueva clase llamada *PruebaImportaciones* que no estará incluida en ningún paquete. A continuación, debes hacer lo siguiente:

- Importar las 4 clases que vamos a usar (Circulo, Triangulo...)
- Crear un método *main* que cree un objeto de cada clase importada.
- Llame al método *quienSoy* de cada uno de los objetos creados.

13. Abre un nuevo proyecto y llámale U1.E13.PruebaScanner. Debes crear una clase llamada *PruebaScanner* que tenga un método *main* que realice las siguientes acciones:

- Debe crear un objeto Scanner para que pueda leer datos del teclado.
- El programa te pedirá tu nombre y lo guardará en una variable.
- El programa te pedirá tu primer apellido y lo guardará en una variable.
- A continuación, mostrará el texto: "Dime un número entero: " y leerá el número del teclado guardándolo en una variable.
- Se mostrará el texto: "Dime otro número entero: " y leerá el número del teclado guardándolo en otra variable.
- A continuación, el programa mostrará un texto con tu nombre y primer apellido.
- Por último, debes imprimir una cadena que diga: "Los números leídos son Y y Z. Su suma es *resSuma* y su producto es *resProducto*" donde Y y Z se reemplazarán por los números leídos y *resSuma* y *resProducto* por los resultados correspondientes a la suma y la multiplicación de ambos.



14. Abre un nuevo proyecto U1.E14.Expresiones. Debes crear una clase llamada PruebaExpresiones que tenga un método *main* que realice las siguientes acciones:

- Debe crear un objeto Scanner para que pueda leer datos del teclado.
- El programa mostrará un texto que indique “Dame el número a”. Entonces se leerá por teclado un número real (*double*) se guardará en la variable *a*.
- Hará lo mismo tres veces más, pidiendo números reales para las variables *b*, *c* y *d*.
- Finalmente, el programa debe realizar los siguientes cálculos y mostrar los resultados:

$$\frac{a+b*38}{c*d} \quad a + \frac{30-b}{c+d} \quad b * a * \frac{a+b}{b/d} \quad \frac{\frac{a+50}{b}}{\frac{c}{4+d}}$$

15. Abre un nuevo proyecto U1.E15.Factura. Debes crear una clase llamada PruebaFactura que tenga un método *main* que realice las siguientes acciones:

- Debe crear una constante llamada IVA cuyo valor es 21.
- Debe crear un objeto Scanner para que pueda leer datos del teclado.
- El programa mostrará un texto que indique “Dime el precio de un pen-drive”. Entonces se leerá por teclado un número real (*double*) se guardará en la variable *precio*.
- El programa mostrará un texto que indique “Dime el número de pen-drives que quieres comprar”. Entonces se leerá por teclado un número entero (*int*) se guardará en la variable *numPen*.
- A continuación, debe simular el resultado una factura imprimiendo lo siguiente:

FACTURA JAMAZON

Precio unitario: *precio* €

Número artículos: *numPen*

Subtotal: *resultadoSinIva* €

-----

Total (IVA incluido): *resultadoConIva* €