Desarrollo de clases

MÉTODOS CONSTRUCTORES

1. Crear una clase llamada Persona con atributos, nombre y numeroDeLaSuerte, y un método saludar que muestre “Hola, soy <nombre> y mi número de la suerte es <numeroDeLaSuerte>”. Cada vez que se crea una Persona se le ha de poder poner un nombre. Si no se especifica, la Persona se “bautizará” a sí misma con el nombre “Anónimo”. Cada objeto Persona escogerá su propio número de la suerte (un número entre 1 y 10 al azar) en el momento en el que se crea. Hacer un programa que, en un bucle infinito me permita introducir nombres de Persona, cree un objeto con ese nombre y le pida que salude. Si se introduce el nombre “.”, se creará una Persona anónima, y si se introduce el nombre “s” el programa finalizará.
2. Hacer una clase A, y un programa que cree en bucle n objetos de dicha clase (siendo “n” un número al azar entre 1 y 100). Crear un método estático **numeroDeObjetos():int** en A que me devuelva el número de objetos de la clase A creados hasta el momento. Utilizarlo después del bucle para comprobar que ha funcionado bien (comparándolo con el número al azar generado, y viendo que coinciden).

AGREGACIÓN de OBJETOS

3. Crear una clase llamada Naranja que tenga un atributo de tipo entero y privado llamado “peso” y dos métodos: un constructor parametrizado que admita un argumento entero para darle un peso inicial a la naranja y un método llamado comer( ) que imprima por pantalla “mmm..qué rica naranja”.
4. Crear una clase llamada Limon similar a la anterior pero cuyo método comer( ) imprima por pantalla “fff..qué limón más ácido”.
5. Crear una clase llamada CestaCompra que tenga un atributo n de clase Naranja y un atributo l de clase Limon (que no sean privados). Crear un constructor sin parámetros para la clase CestaCompra que cree un nuevo objeto de la clase Naranja de 50 gramos y un nuevo objeto de la clase Limon de 30 gramos y se los asigne a sus atributos n y l respectivamente.
   La clase CestaCompra deberá tener un método que se llame comerFruta( ) que invoque a los métodos comer( ) de l y de n.
4. Crear un programa para probar la clase CestaCompra creada anteriormente. El programa deberá crear un objeto de la clase CestaCompra y posteriormente deberá invocar al método comerFruta(), obsérvese lo que sale por pantalla.
5. Encontrar un método alternativo para comerse las dos frutas de la cesta sin necesidad de llamar al método comerFruta( ) de CestaCompra. CONSEJO: Comerse cada fruta por separado (desde el programa principal) invocando a sus métodos comer( ) respectivos.

6. Cambiar los atributos n y l de CestaCompra de manera que sean privados (private). Recompilar y observar qué ocurre al ejecutar el Ejercicio 5 de nuevo. Razónese dicho comportamiento y cómo nos podríamos comer las frutas.
7. Sobrecargar el constructor de CestaCompra con uno nuevo parametrizado que admita dos parámetros de tipo entero (que serán el peso de la naranja y limón que contiene la cesta de la compra).
    a. Añadir un método getPeso( ) a la clase Naranja y otro getPeso( ) a la clase Limon que le devuelva al invocante el peso de cada fruta.
    b. Añadir un método getPesos( ) en la clase CestaCompra que muestre los pesos de sus atributos n y l.
8. Crear un programa que cree varios objetos CestaCompra con distintos pesos para sus naranjas y limones respectivos, y a continuación muestre por pantalla el peso de la naranja y el limón de cada una de las cestas que hemos creado.

ELEMENTOS ESTÁTICOS, CONSTRUCTORES, SOBRECARGA, HERENCIA, SOBRESCRITURA

9. Crear las clase “Lineas” y deducir el comportamiento de sus métodos para que al ejecutar el método “main” de la clase Prueba salga por pantalla lo siguiente:

```
**
****
******
    **
  ****
******
```

```
public class Prueba {
   public static void main(String[] args) {
         Lineas l=new Lineas(“*”);
         l.linea(2);
         l.linea(4);
         l.linea(6);
         l.linea(4,2);
         l.linea(2,4);
         l.linea(0,6);
   }
}
```

10. Escribir el código “main” de una clase “Prueba” para que, al ejecutarlo y utilizando la clase “Letras” salga por pantalla la cadena “Carlos es inaudito” (sin utilizar el comando “println” en “main” y sin utilizar herencia)

```
public class Letras {
      private String cadena;
      public Letras(String cadena) {this.cadena=cadena;}
      public void setCadena(String cadena) {this.cadena=cadena;}
      public void p(String cadena) {
            System.out.print(this.cadena + ”a” + cadena);
      }
      public static void q() {
            System.out.print(“ es ”);
      }
}
```

11. Escribir el código de “main” de una clase “Prueba” para que salga por pantalla “Pérez Manzano Santos Borreguero”, con las siguientes restricciones:

- No se puede utilizar “print” ni “println”
- Se puede utilizar la clase “Persona”
- No se pueden utilizar más clases, aparte de Persona (no utilizar herencia)

```
public class Persona {
private String nom;
private String ape1;
private String ape2;
public Persona(String nom, String ape1, String ape2) {
	this.nom=nom;
	this.ape1=ape1;
	this.ape2=ape2;
}
public static void mostrarApellidos(Persona padre, Persona madre) {
	System.out.println(
		padre.ape1+” ”+madre.ape1+” ”+
		padre.ape2+” ”+madre.ape2
		);
}
}
```