ESCUELAS SALESIANAS Mª AUXILIADORA
C/ Santa Mª Mazzarello, 1
41005· Sevilla
Tlf: 954 63 23 47
www.nervion.salesianas.com

HERRAMIENTAS DE EVALUACIÓN

PROGRAMACIÓN		
Curso y grupo: 1° DAM	**Módulo:**	**0373**
Fecha de entrega IE: 10/12/2018	**UT 2 y UT4**	**Realizar aplicaciones sencillas utilizando los conceptos de la programación orientada a objetos.** **Crear aplicaciones estructuradas en clases utilizando las características de la programación orientada a objetos.**
	I.E.	**4.1.**

Resultados de aprendizaje:

RA2. Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos.
RA4. Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.

Metodologia

Explicación teórica y ejercicios prácticos.

Equipos y materiales:

Equipos informáticos (IDE Eclipse) apuntes de clase, manuales e Internet.

Temporalización:

3 horas.

Criterios de evaluación:

2.1. Conoce las características de la programación orientada a objetos
2.2. Conoce la estructura básica de una clase y sabe diferenciar entre un objeto y una clase
2.3. Sabe crear y utilizar objetos, identificando las propiedades, los métodos y constructores.
2.4. Diseña y programa métodos que acepten parámetros.
2.5. Organiza clases en paquetes y sabe utilizar los mismos.
4.1. Comprende en profundidad el concepto de clase.
4.2. Diseña e implementa la estructura y miembros de una clase.
4.3. Conoce e implementa las distintas características de los métodos de una clase.
4.4. Conoce y aplica la encapsulación, control de acceso y visibilidad de los miembros de una clase.
4.5. Conoce el concepto de constructor y finalizador.
4.6. Conoce y aplica el concepto de herencia en la resolución de problemas.
4.7. Comprende el concepto de interface y su aplicación en Java.
4.8. Conoce y aplica los conceptos de polimorfismo en Java.

ESCUELAS SALESIANAS Mª AUXILIADORA
C/ Santa Mª Mazzarello, 1
41005· Sevilla
Tlf: 954 63 23 47
www.nervion.salesianas.com

Actividades

1. (1 punto) Justifica cuál sería el resultado de la ejecución del siguiente código:

```
package paquete1;

public class Numero {

   private int valor;

   public Numero(int valor){
         this.valor = valor;
   }

   public int getValor() {
      return valor;
   }

   public void setValor(int valor) {
      this.valor = valor;
   }
}
```

```
package paquete1;

public class Envoltorio {

  private Numero n;

  public Envoltorio(Numero n){
      n.setValor(0);
  }

  public Numero getNumero(){
      return n;
  }

  public static void main(String[] args) {
      Numero n = new Numero(3);
      Envoltorio e = new Envoltorio (n);

      System.out.println(n.getValor());
      System.out.println(e.getNumero().getValor());
  }
}
```

2. (0,5 puntos) Indica qué constructores deben estar definidos en la clase A para que el siguiente código sea correcto. Justifica la respuesta.

```
public class B extends A {

    public B(int j) {
    }

    public B(int j, int k) {
      super(j, k);
```

```
        }

    }
```

3. (1 punto) Señala los tres errores de compilación del siguiente código y corrígelo.

```
package prueba;

public class A {

      int at1;

      public A(int v){

            at1 = v;
      }
}
```

```
package prueba2;

public class B extends A {

      private int at2;

      public B(int v){

            at1 = v;
            at2 = v;

      }

}
```

4. (2 puntos) Diseña una clase Dirección con los atributos calle, número, piso y ciudad. A continuación, diseña una clase Empleado con los atributos nombre (de tipo String), salario (de tipo int) y dirección (de tipo Dirección).

   Para probar la funcionalidad de las clases creadas crea una clase Test donde crees 3 direcciones y luego crea tres empleados, asignándoles una de las direcciones anteriormente creadas. Por último, crea un método que muestre los datos de cada empleado creado:

   EMPLEADO 1:
   Nombre: XXX
   Salario: XXX
   Dirección:
   Calle: XXX
   Número: XXX
   Puerta: XXX
   Ciudad: XXX
   EMPLEADO 2:
   Nombre: ……..

5. (2,5 puntos)Realizaremos un programa en java con la siguiente distribución de clases:
   Crear una clase Vehículo caracterizada por:
   Los atributos privados:
   - Matrícula, color del vehículo, número de ruedas, cilindrada, potencia

   Los métodos constructores:
   - El primero permite crear un vehículo a partir de la matrícula, color y número de ruedas.
   - El segundo permite crear un vehículo a partir de los datos señalados en el primer constructor y de la cilindrada.

- El tercero permite crear un vehículo a partir de los datos señalados en el segundo constructor y de la potencia.

Métodos getter/setter que permitan:

- Asignar la cilindrada
- Asignar la potencia
- Un método para cada uno de los atributos que permitan devolver su valor.

Crear una clase Coche que derive de la clase Vehículo caracterizada por:

Los atributos privados:

- numPuertas

Los métodos constructores que invocarán al constructor de la clase padre siempre que sea posible. (Un coche tiene 4 ruedas).

- El primero permite crear un coche con la matrícula y color.
- El segundo permite crear un coche a partir de los datos señalados en el primer constructor y de la cilindrada.
- El tercero permite crear un coche a partir de los datos del segundo constructor y de la potencia.
- El cuarto permite crear un coche a partir de los datos del tercer constructor y del número de puertas.

Métodos getter/setter que permitan:

- Un método que permita asignar el numPuertas
- Un método que permita devolver el numPuertas

Crear una clase Moto que derive de la clase Vehículo caracterizada por:

Los atributos privados:

- numPlazas

Los métodos constructores, que invocarán al constructor de la clase padre siempre que sea posible. (Una moto tiene 2 ruedas)

- El primero permite crear una moto con la matrícula y color.
- El segundo permite crear una moto a partir de los datos señalados en el primer constructor y de la cilindrada
- El tercero permite crear una moto a partir de los datos del segundo constructor y de la potencia.
- El cuarto permite crear un coche a partir de los datos del tercer constructor y del número de plazas.

Crear una clase PruebaVehiculo con un método principal.

- Crear un objeto de tipo Coche de acuerdo con los datos siguientes: matricula=”0000BBB” color “gris plata” y el número de puertas 3.
- Crear una variable de tipo Moto de acuerdo con los datos siguientes: Matricula=”2222BBB” color “negro” y el número de plazas es 2.
- Visualizar los datos del coche y de la moto.

6. (3 puntos) Diseñar una clase coche con el atributo matrícula, marca y modelo. Diseñar el constructor, el getter y setter y algún método para mostrar todos sus atributos. Siempre que se cree un coche deberá de ser con los tres atributos.

   Diseñar una clase plaza de aparcamiento con los siguientes atributos:

   - número de plaza – que contendrá un número de plaza,
   - un atributo coche que almacenará un objeto coche cuando aparque el mismo, de forma que cuando este libre se le asignará el valor null.

- Un atributo que permita saber si la plaza está disponible o no
- Otro atributo que permitirá contabilizar el número de coches que han aparcado en dicha plaza

Diseñar el constructor asignándole como número de plaza un número y el resto de los valores se cargarán con el valor por defecto.

Crear los métodos getter y setter, así como un método que permita visualizar los valores de los atributos.

Crear un método que permita mostrar un menú con las siguientes opciones

MENU PLAZA APARCAMIENTO
1. Aparcar coche
2. Sacar coche
3. Ver coche aparcado
4. Salir aplicacion

OPCION:

Crear un método que permita aparcar un coche, en cuyo casi, y siempre que la plaza esté disponible se pedirán por teclado los datos del coche (matricula, marca y modelo) para crear el objeto y se procederá a aparcar el coche, modificando los atributos oportunos de la plaza de aparcamiento. Se deberá de avisar que la plaza ya está ocupada por otro coche en el caso de que se seleccione esta opción y la plaza no esté disponible indicarlo con el mensaje oportuno PLAZA YA OCUPADA POR OTRO COCHE. Si el coche se ha podido aparcar se deberá mostrar EL COCHE SE HA APARCADO.

Crear un método que permita sacar un coche de la plaza mostrando en este caso PLAZA LIBRE EL COCHE…. HA SALIDO DE LA PLAZA y modificar el valor de los atributos. En el caso de que querer sacar un coche de la plaza y esta no esté ocupada indicar el mensaje ERROR, NO HAY COCHE EN LA PLAZA DE APARCAMIENTO LA PLAZA ESTÁ LIBRE

Crear un método que permita mostrar el coche que está aparcado debiendo mostrar en este caso, la información del coche que está aparcado o en su caso LA PLAZA ESTA LIBRE.

Cuando se desee finalizar el programa se deberá mostrar el estado de la plaza de aparcamiento, con la información del coche si lo hay y el total de coches que han aparcado en ella.

Crear una clase MainGestionPlaza que permita gestionar dicha plaza a partir del menú.

Para la realización de los ejercicios se tendrá en cuenta para su calificación las siguientes valoraciones:

- ***El control de excepciones aplicado en el código.***
- ***Los comentarios en el código.***
- ***La mejor utilización posible de las características de la Programación Orientada a Objetos (creación de clases, instanciación de objetos, accesibilidad de los miembros de una clase, miembros estáticos, creación adecuada de paquetes y librerías, encapsulamiento de las aplicaciones, sobrecarga de constructores, herencia, utilización de operadores this y super …)***

Una vez realizada la tarea se realizará la exportación de la carpeta del proyecto de Eclipse (el nombre del proyecto será Torres_Molina_Manuel_PROGUT2y4_I.E.4.1, donde vendrán todas las actividades desarrolladas en Java. El envío se realizará a través de la plataforma de la forma establecida para ello.