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: 3/12/2018	**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.	**2.1.**

Resultados de aprendizaje:

RA1. Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos.
RA3. 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:

16 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.

Actividades

1. (0,5 puntos) Crear una clase para datos de películas

   Escribe una clase que represente una película en DVD de nombre ***DVDCine*** con los atributos necesarios para mostrar la ficha que se recoge en el apartado segundo. Esta clase contará con un constructor que admite como argumentos todos los atributos de la clase.

   Escribe los siguientes métodos para la clase DVDCine:

   - ***muestraDVDCine***: Este método retorna una descripción completa de la película, por ejemplo:

     UN FINAL MADE IN HOLLYWOOD (HOLLYWOOD ENDING) De: Woody Allen
     Con: Woody Allen y George Hamilton

     Comedia - 114 min.

     Resumen: Los Oscars ganados en el pasado por el ex-genio del cine …

   - ***esThriller***: Este método retorna cierto (true), si la película pertenece a este género cinematográfico.

   - ***tieneResumen***: Retorna cierto si la película tiene el resumen escrito.

   - ***muestraDuracion***: Retorna la duración con el formato: "114 min ."

   Escribir una aplicación que solicite los datos de una película, genere un objeto DVDCine y muestre éste con el formato del apartado segundo. Probar también los métodos esThriller y tieneResumen.

2. (0,5 puntos) Crear una clase de nombre ***Vehiculo*** que contenga como atributos el modelo de tipo String, la potencia de tipo double y la tracción a las cuatro ruedas cRuedas de tipo boolean. El constructor de la clase admitirá como argumento el modelo y tendrá como métodos de tipo get y set para la potencia y para la tracción a las cuatro ruedas. La clase contará con el método imprimir que retorna los datos de cada vehículo y si tiene tracción a las cuatro ruedas.

3. (0,5 puntos) Crea una clase ***Cuenta*** (bancaria) con atributos para el número de cuenta (un entero largo), el DNI del cliente (otro entero largo), el saldo actual y el interés anual que se aplica a la cuenta (porcentaje). Define en la clase los siguientes métodos:

   - Constructor por defecto y constructor con DNI, saldo e interés
   - Accedentes y mutadores. Para el número de cuenta no habrá mutador.
   - ***actualizarSaldo()***: actualizará el saldo de la cuenta aplicándole el interés diario (interés anual dividido entre 365 aplicado al saldo actual).
   - ***ingresar(double)***: permitirá ingresar una cantidad en la cuenta.

- ***retirar(double)***: permitirá sacar una cantidad de la cuenta (si hay saldo).
- Método que permita mostrar todos los datos de la cuenta.

El número de cuenta se asignará de forma correlativa a partir de 100001, asignando el siguiente número al último asignado.

4. (0,5 puntos) Desarrolla una clase Cafetera con atributos *_capacidadMaxima* (la cantidad máxima de café que puede contener la cafetera) y *_cantidadActual* (la cantidad actual de café que hay en la cafetera). Implementa, al menos, los siguientes métodos:

   - Constructor predeterminado: establece la capacidad máxima en 1000 (c.c.) y la actual en cero (cafetera vacía).
   - Constructor con la capacidad máxima de la cafetera; inicializa la cantidad actual de café igual a la capacidad máxima.
   - Constructor con la capacidad máxima y la cantidad actual. Si la cantidad actual es mayor que la capacidad máxima de la cafetera, la ajustará al máximo.
   - Accedentes y mutadores.
   - ***llenarCafetera():***hace que la cantidad actual sea igual a la capacidad.
   - ***servirTaza(int):*** simula la acción de servir una taza con la capacidad indicada. Si la cantidad actual de café "no alcanza" para llenar la taza, se sirve lo que quede.
   - ***vaciarCafetera():*** pone la cantidad de café actual en cero.
   - ***agregarCafe(int):*** añade a la cafetera la cantidad de café indicada.

5. (0,5 puntos) Crea una clase **Empleado** que modele la información que una empresa mantiene sobre cada empleado: NIF, sueldo base, pago por hora extra, horas extra realizadas en el mes, tipo (porcentaje) de IRPF, casado o no y número de hijos. La clase debe contemplar accedentes y mutadores para todos los atributos. Al crear los objetos se podrá proporcionar, si se quiere, el número de DNI. Los demás servicios que deberán proporcionar los objetos de la clase serán los siguientes:

   - Cálculo y devolución del complemento correspondiente a las horas extra realizadas.
   - Cálculo y devolución del sueldo bruto.
   - Cálculo y devolución de las retenciones (IRPF) a partir del tipo, teniendo en cuenta que el porcentaje de retención que hay que aplicar es el tipo menos 2 puntos si el empleado está casado y menos 1 punto por cada hijo que tenga; el porcentaje se aplica sobre todo el sueldo bruto.
   - ***println():*** visualización de la información básica del empleado.
   - ***printAll():*** visualización de toda la información del empleado. La básica más el sueldo base, el complemento por horas extra, el sueldo bruto, la retención de IRPF y el sueldo neto.
   - ***copia():*** clonación de objetos.

6. (0,5 puntos) Confeccionar una clase **Persona** que tenga como atributos el nombre y la edad. Definir como responsabilidades un método que cargue los datos personales y otro que los imprima.
   Plantear una segunda clase **Empleado** que herede de la clase Persona. Añadir un atributo sueldo y los métodos de cargar el sueldo e imprimir su sueldo.

Definir un objeto de la clase Persona y llamar a sus métodos. También crear un objeto de la clase empleado y llamar a sus métodos.

7. (0,5 puntos) Implementar la clase Asignatura que represente el nombre de una asignatura y la nota correspondiente obtenida. Las operaciones son:

   - Constructor que acepte como parámetros el nombre de la asignatura y la nota obtenida.
   - Métodos para modificar la nota (setNota) y para consultar la nota (getNota).
   - Método que nos devuelva "Aprobado" si la nota es mayor o igual a 5 o "Suspenso" si la nota es menor que 5.
   - Método para consultar el nombre de la asignatura.

8. (0,5 puntos) Implementar la clase Alumno que incluya la Asignatura a la que el alumno ha asistido. Además de incluir los atributos nombre y edad. Las operaciones disponibles sobre el alumno son:

   - Constructor que acepte como parámetro el nombre del alumno y edad.
   - Métodos para modificar el nombre (setNombre) y para consultarlo (getNombre).
   - Métodos para modificar y consultar la edad.
   - Implementar la clase test para hacer uso de las clases Alumno y Asignatura.

   Crear 3 alumnos (Tres instancias de la clase Alumno) con sus respectivos nombre y edad.

   Para cada alumno establecer sus asignaturas y la nota obtenida.

   Imprimir en pantalla:

   - Nombre del alumno.
   - Edad.
   - Asignaturas que cursó:
     - Nombre de la asignatura.
     - Nota obtenida.
     - Si es una asignatura aprobada o no.

9. (0,5 puntos) Construya una clase Persona que tiene los siguientes atributos, apellidos, nombres, sexo y DNI, incluya sus constructores y un método que permita visualizar la información de los atributos.

   Luego construya la subclase Docente de la clase Persona que tiene los atributos: Código, categoría, Estudios de Postgrado, horas de clase y pago por hora. Construya su constructor y los siguientes métodos que permitan:

   Calcular el Pago Parcial que se calcula de: número de horas * pago por Hora

Luego construya una subclase hija Estudiante de la clase Persona que tendrá los atributos: Código, Categoría, Pago de pensiones y colegio de procedencia Construya su constructor y los siguientes métodos que permitan:

Calcular el descuento sobre el pago parcial de pensiones de acuerdo al promedio ponderado como sigue a continuación

Rango del Prom Ponderado	% de descuento sobre el pago de pensión
Menor a 13	0%
Entre 13 y 16	8%
Mayor a 16	15%

10. (0,5 puntos) El restaurante andaluz de Casa Conde, cuya especialidad son las papas con chocos, nos pide diseñar un método con el que se pueda saber cuántos clientes pueden atender con la materia prima que tienen en el almacén. El método recibe la cantidad de papas y chocos en kilos y devuelve el número de clientes que puede atender el restaurante teniendo en cuenta que por cada tres personas, Israel utiliza un kilo de papas y medio de chocos.

    Desarrolla un programa creando una clase que permita almacenar los kilos de papas y chocos del restaurante. Implementa los siguientes métodos:
    - public void ***addChocos***(int x). Añade x kilos de chocos a los ya existentes.
    - public void ***addPapas***(int x). Añade x kilos de papas a los ya existentes.
    - public int ***getComensales***(). Devuelve el número de clientes que peude atender el restaurante (este es el método anterior).
    - public void ***showChocos***(). Muestra por pantalla los kilos de chocos que hay en el almacén.
    - public void ***showPapas***().Muestra por pantalla los kilos de papas que hay en el almacén.

11. (0,5 puntos) Se quiere definir una clase Triangulo que represente al polígono Triangulo.

    Atributos
    double long_lado1;
    double long_lado2;
    double long_lado3;

    Métodos
    public Triangulo (double l1, double l1, double l1)
    /* Constructor sobrecargado que inicializa los atributos. */
    public boolean compareTo_Triangulos(Triangulo a, Triangulo b)
    /* Método que que permite determinar si dos triangulos son o no iguales. */

    public boolean compareTo_VTriangulos(Triangulo v[ ])
    /* Método que permite determinar si un conjunto de Triangulos son iguales. */

public int tipo_Triangulo()
/* Método que indica si el triángulo es equilátero (1), isósceles (2), escaleno (3).*/

Por la longitud de sus lados, los triángulos se clasifican en:
- **Triángulo equilátero**: si sus tres lados tienen la misma longitud.
- **Triángulo isósceles**: si tiene dos lados de la misma longitud.
- **Triángulo escaleno**: si todos sus lados tienen longitudes diferentes.

Probar en una clase con el método main las distintas funcionalidades de la clase Triángulo.

12. (0,5 puntos) Desarrolle una aplicación en Java que determine el sueldo bruto para cada uno de los tres empleados de una empresa. La empresa paga la tarifa normal en las primeras 40 horas de trabajo de cada empleado, y paga tarifa y media en todas las horas trabajadas que excedan de 40.
El programa creará 3 objetos (uno para cada empleado) y se le pedirá al usuario que rellene la información para cada empleado en el constructor.
Por cada empleado se almacenará su nombre, el número de horas que trabajó, y la tarifa que cobra por una hora de trabajo.
Para probar la clase Empleado, crea una clase Test con un método que determine y muestre el sueldo bruto de cada empleado.

Ejemplo:
Pepe trabajó 42 horas, cobra 20 euros la hora por lo que le corresponde un sueldo de 860 euros.

13. (0,75 puntos) Crear una clase de nombre **Alimento** cuyos objetos representen alimentos. Éstos, serán identificados por una descripción alfanumérica que representa el nombre y tendrán además los atributos siguientes:

    - Contenido en lípidos expresado en tanto por ciento.
    - Contenido en hidratos de carbono expresado en tanto por ciento.
    - Contenido en proteínas expresado en tanto por ciento.
    - Si es o no de origen animal.
    - Contenido en vitaminas expresado en los códigos A alto, M medio y B bajo.
    - Contenido en minerales expresado en los códigos A alto, M medio y B bajo.

    La clase tiene dos constructores: uno que admite como argumentos el nombre del alimento, y otro que admite todos los atributos.

    La clase tiene los siguientes métodos:
    - ***esDietetico*** retorna cierto si el alimento contiene menos del 20% de lípidos y el contenido en vitaminas no es bajo.
    - ***muestraAlimento*** retorna una descripción del alimento.

14. (0,75 puntos) Diseñar una clase llamada Marciano con los siguientes atributos:
    - **vivo** de tipo boolean y false por defecto
    - **nombre** de tipo string

- **numMarcianos** que nos permitirá contabilizar el número de marcianos vivos que hay en cada momento.

Se crearán los siguientes métodos:

- Un método constructor que pasará como parámetro el nombre del marciano y se encargará de realizar las siguientes acciones:
  - Cambiar el estado del atributo vivo y ponerlo a true.
  - Asignar el nombre al atributo nombre.
  - Incrementar el número de marcianos.
  - Invocar al método nace.
- Un método llamado **nace** que nos permitirá visualizar el mensaje: "Hola, he nacido y soy el marciano ……"
- Un método llamado **muere** que realiza las siguientes acciones:
  - Si el marciano está vivo, cambiar el valor del atributo vivo, actualizar el valor del atributo numMarcianos y visualizar el mensaje "El marciano …….. ha muerto".
  - Si el marciano está muerto, visualizar el mensaje "El marciano ya está muerto".
- Un método llamado **estaVivo** que nos permitirá saber si un marciano está vivo o no.
- Un método llamado **cuentaMarcianos** que nos dirá cuántos marcianos vivos hay en ese momento.

Diseñar una clase llamada Marte donde estará el método main, que permita crear y matar marcianos:

- Deberá crear 3 marcianos et1, et2 y et3.
- Después deberá matar al segundo marciano (et2).
- A continuación deberá crear un nuevo marciano et4.
- Intentará matar al marciano et2.
- Cada vez que se cree un marciano deberá aparecer el mensaje "Hola, he nacido y soy el marciano ……", y mostrar el número de marcianos vivos en ese momento.
- Al final, deberá mostrar la situación en la que está cada uno de los marcianos.

15. (0,75 puntos) Se pretende desarrollar una aplicación que simule el funcionamiento de un cajero automático. Primeramente, se debe crear una clase llamada Cuenta, que gestione las operaciones sobre la cuenta. Además de los constructores y campos que se estimen necesarios, la clase contará con los métodos:

    - void ***ingresar(float c)*** //Agrega al saldo de la cuesta la cantidad recibida.
    - void ***extraer(float c)*** //Descuenta del saldo la cantidad recibida. Tras la llamada a este método, el saldo podrá quedar en negativo.
    - float ***getSaldo()*** //Devuelve el saldo actual

    Por otro lado, existirá una clase con el método main encargada de la captura y presentación de datos, y de la gestión de la cuenta. Al iniciarse la aplicación se mostrará el siguiente menú:

    1.- Crear cuenta vacía.
    2.- Crear cuenta con saldo inicial.
    3.- Ingresar dinero.

4.- Sacar dinero.
5.- Ver saldo.
6.- Salir.

La opción 1 crea un objeto Cuenta con saldo 0, la opción 2 solicita una cantidad y crea un objeto Cuenta con este saldo inicial. En la opción 3 se solicita una cantidad y la ingresa en el objeto creado en las opciones 1 o 2 (debe haber pasado antes por estas opciones), mientras que en la opción 4 se solicita una cantidad y la extrae del objeto creado en las opciones 1 o 2 (también debe haber pasado antes por estas opciones). Finalmente, la opción 5 muestra el saldo, mientras que la 6 finaliza el programa.
El menú vuelve a presentarse en pantalla mientras no se elija la opción de salir.

16. (0,75 puntos) Un videojuego tiene **Personajes**. Cada personaje tiene un nombre (String) y un nivel propio de energía (int). Además implementan el método alimentarse, que recibe por parámetro una cantidad de energía(int) con el que incrementa el nivel propio de energía. Los personajes pueden ser:

    - ***Guerreros***: tienen además un arma (String). Al momento de la instanciación reciben su nombre, arma y nivel propio de energía inicial. Los guerreros tienen un método combatir que recibe por parámetro la cantidad de energía a gastar en el ataque, la cual es descontada de su nivel propio de energía. El método combatir retorna el arma y la cantidad de energía del ataque concatenados.

    - ***Magos***: tienen además un poder (String). Al momento de la instanciación reciben su nombre y poder. Los magos son siempre creados con un nivel propio de energía igual a 100. Proveen un método encantar, que disminuye en 2 unidades el nivel propio de energía y que retorna el poder del mago.

    Implementar una clase test que instancie objetos de Guerreros y Magos y pruebe todos los métodos de los mismos para comprobar que el programa funciona correctamente.

17. (1 punto) Realiza un programa en JAVA que constará de las siguientes clases: Ordenador, Servidor (hereda de Ordenador), Portátil (hereda de Ordenador) y Test contiene el método main y otros métodos estáticos.

    - La superclase ***Ordenador*** contiene los siguientes atributos: cantidad de memoria RAM, capacidad del disco duro, modelo de procesador, modelo de tarjeta gráfica y precio. A la hora de que el usuario cree un objeto, en la clase Test, deberemos controlar que la capacidad del disco duro sea múltiplo de 5, el modelo de procesador lo permitirá elegir entre una lista, y el precio siempre sea mayor que 0.

    - La subclase ***Servidor*** contiene los siguientes atributos: tamaño del monitor, modelo de teclado y modelo de ratón. A la hora de que el usuario cree un objeto, en la clase Test, deberemos controlar que el tamaño del monitor sea mayor que 14. (en caso de que el usuario no lo cumpla deberemos volvérselo a pedir, tantas veces como sea necesario.)

- La subclase ***Portátil*** contiene los siguientes atributos: marca, tamaño de pantalla y peso.

- La clase **Test** contendrá los métodos estáticos necesarios para controlar que el usuario introduzca los datos correctos. Además, contiene el método main donde se crearán 2 objetos de la clase Servidor y 2 de la clase Portátil mediante un constructor que reciba los parámetros. También se crearán 1 objetos de la clase Servidor y 1 de la clase Portátil mediante el constructor por defecto, por lo que necesitarás llamar posteriormente a los métodos "establecer…" para asignar los valores que te indique el usuario a los atributos del objeto.

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

- ***El control de excepciones aplicado en el código.***
- ***El código debe estar documentado utilizando las anotaciones correctas.***
- ***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.2.1, donde vendrán todas las actividades desarrolladas en Java. El envío se realizará a través de la plataforma de la forma establecida para ello.