Trabajo Practico №3
Paradigmas de Programación

Dr. Pablo Javier Vidal Unidad 2

Introducción

Ejercicio 1.

Defina qué es un objeto. De ejemplos concretos

1. Defina qué es una clase. De ejemplos concretos.
2. ¿Cuál es la diferencia entre un objeto y una clase?
3. ¿Todos los lenguajes de objetos poseen clases? ¿Cualés no? ¿Qué ventajas y desventajas tienen entre sí?

Ejercicio 2.

Para cada definición de clase utilice un diagrama de clase para definir los atributos y los métodos.

1. Perro, ¿es un objeto o una clase? Justificar.
2. Definir la clase **Auto**. Definir 6 atributos y al menos 10 métodos
3. Declarar una clase llamada **Circunferencia** que tenga como atributo el radio de tipo real y los siguientes métodos:
   a) Calcular el área.
   b) Calcular el perímetro.
   c) Método para modificar el radio
   d) Método para consultar el valor del radio
4. Crear una clase **Contador** con los métodos para incrementar y decrementar el contador. La clase debe tener un método que permita mostrar el valor actual del contador, volver a 0 el contador, definir el incremento del contador e incrementar el contador.
5. Crear una clase **Libro** con los métodos préstamo, devolución y transformarATexto. La clase métodos de acceso a las propiedades.
6. Crear una clase **Fecha**. La clase debe contener los métodos de acceso a los atributos. También deberá comprobar una fecha dada es correcta y modificar la fecha actual que tiene la clase definida.
7. Definir la clase **NumeroComplejo**. Definir sus atributos y métodos
8. Realizar una clase llamada **Persona** que tenga los siguientes atributos: nombre, edad, DNI, sexo ('H' hombre, 'M' mujer, 'O' otro), peso y altura. Si el alumno desea añadir algún atributo, puede hacerlo.
9. Defina la clase **Electrodoméstico** y proporcione 3 instancias de esta clase. Informar los valores de los atributos de cada objeto

10. Definir la clase **Cuenta**. La misma deberá tener como mínimo los atributos *numero_ cuenta* y *saldo*. Como operaciones deberá tener consultarSaldo() y debitar(unValor) como mínimo. Definir el tipo de dato, de los métodos y de los parámetros en caso de ser necesario.

11. Defina una clase que permita definir un objeto rectángulo y un triángulo. ¿El círculo pertenecería a su clase definida? ¿Porque? Justifique.

12. Definir 2 clases las cuales estén relacionadas con su día a día. Las mismas deberán estar compuestas por al menos 2 atributos y 4 métodos que tengan directa relación con la clase. Posteriormente, definir dos instancias de cada clase.

    Nota: las clases definidas en este practico no son validas como parte del ejercicio.

Ejercicio 3.

1. ¿Cuál es la diferencia entre mensaje y método?

2. ¿Cuál es la diferencia entre enviar un mensaje e invocar una función en el paradigma estructurado?

3. ¿Qué relación tiene la separación mensaje/método con el dynamic binding?

4. $+ - */$ ¿Son mensajes, métodos o funciones? Justificar

Clases y métodos

Para cada clase definida, implementar al menos dos constructores y los getters y setters necesarios.

Ejercicio 4.

1. Crea una clase *Cuenta* (bancaria) con atributos para el número de cuenta (un entero), 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, número de cuenta e interés
    - 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 nos permita mostrar el saldo de la cuenta
    - Método que nos permita mostrar todos los datos de la cuenta.

Ejercicio 5.

1. Crear una clase *Rectangulo* que modele rectángulos por medio de cuatro puntos (los vértices). Dispondrá de dos constructores: uno que cree un rectángulo partiendo de sus cuatro vértices y otro que cree un rectángulo partiendo de la base y la altura, de forma que su vértice inferior izquierdo esté en (0,0). La clase también incluirá un método para calcular la superficie y otro que desplace el rectángulo en el plano.

Ejercicio 6.

1. Crea una clase Fecha con atributos para el día, el mes y el año.

   Incluir, al menos, los siguientes métodos:

   - Constructor predeterminado con el 1-1-1900 como fecha por defecto.
   - Constructor parametrizado con día, mes y año.
   - leer(): pedirá al usuario el día (1 a 31), el mes (1 a 12) y el año (1900 a 2050).
   - bisiesto(): indicará si el año de la fecha es bisiesto o no.
   - diasMes(int): devolverá el número de días del mes que se le indique (para el año de la fecha).
   - valida(): comprobará si la fecha es correcta (entre el 1-1-1900 y el 31-12-2050); si el día no es correcto, lo pondrá a 1; si el mes no es correcto, lo pondrá a 1; y si el año no es correcto, lo pondrá a 1900. Será un método auxiliar (privado). Este método se llamará en el constructor parametrizado y en leer().
   - corta(): mostrará la fecha en formato corto (02-09-2003).
   - diasTranscurridos(): devolverá el número de días transcurridos desde el 1-1-1900 hasta la fecha.
   - diaSemana(): devolverá el día de la semana de la fecha (0 para domingo, ..., 6 para sábado). El 1-1-1900 fue domingo.
   - larga(): mostrará la fecha en formato largo, empezando por el día de la semana (martes 2 de septiembre de 2003).
   - fechaTras(long): hará que la fecha sea la correspondiente a haber transcurrido los días que se indiquen desde el 1-1-1900.
   - diasEntre(Fecha): devolverá el número de días entre la fecha y la proporcionada.
   - siguiente(): pasará al día siguiente.
   - anterior(): pasará al día anterior.
   - copia(): devolverá un clon de la fecha.
   - igualQue(Fecha): indica si la fecha es la misma que la proporcionada.
   - menorQue(Fecha): indica si la fecha es anterior a la proporcionada.
   - mayorQue(Fecha): indica si la fecha es posterior a la proporcionada.

Ejercicio 7.

1. Definir la clase *Tiempo*, la cual tendrá la hora, minutos y segundos. Definir los metodos de establecerHora(), establecerMinuto(), establecerSegundos(), imprimirHoraCompleta() y un constructor vacío y otro con la hora, minutos y segundos.

2. Desarrollar una clase *Cancion* con los siguientes atributos:

   - titulo: una variable String que guarda el título de la canción.
   - autor: una variable String que guarda el autor de la canción. y los siguientes métodos:
   - Cancion(String, String): constructor que recibe como parámetros el título y el autor de la canción (por este orden).
   - Cancion(): constructor predeterminado que inicializa el título y el autor a cadenas vacías.
   - dameTitulo(): devuelve el título de la canción.
   - dameAutor(): devuelve el autor de la canción.

- ponTitulo(String): establece el título de la canción.
- ponAutor(String): establece el autor de la canción.

3. Escribe un programa que rellene un array de 30 *doubles* con números aleatorios y luego calcule el promedio. La función *nextDouble*() de la clase *Random* devuelve un número real aleatorio entre 0 y 1

4. Dada una cadena ingresada por pantalla, mostrar la cantidad de vocales que tiene. Ejemplo:
   Entrada: cad = "Hola tu"
   Salida: La cantidad de vocales es 3

Ejercicio 8.

1. Definir una clase Ahorcado (como el juego), la cual deberá contener un vector con la palabra a buscar.

   Definir al menos los siguientes métodos los atributos necesarios:

   - Un método para buscar si la letra es parte de la palabra o no.
   - Un método para informar error o acierto
   - Un método para mostrar cuantas oportunidades quedan.
   - Un método que al pedir ingresar una letra muestre que letras han sido encontradas y cuantas oportunidades quedan.
   - Un método en caso de encontrar la palabra.
   - Un método en caso de no tener más oportunidades.

   El método *main* deberá poder pedir al usuario una letra hasta que el usuario halla gastado todas sus oportunidades o bien hasta que encuentre la letra. Utilizar un vector de caracteres para guardar la palabra
   Un ejemplo de salida puede ser así:

Número de letras (encontradas,faltantes): (3,4)
Número de oportunidades restantes: 3
Estado Actual: _ a _ a _ _ a
Ingrese una letra:
z
Mensaje: NO ES LA LETRA

Número de letras (encontradas,faltantes): (3,4)
Número de oportunidades restantes: 2
Estado Actual: _ a _ a _ _ a
Ingrese una letra:
b
Mensaje: ES UNA LETRA

Número de letras (encontradas,faltantes): (3,4)
Número de oportunidades restantes: 2
Estado Actual: _ a _ a b _ a
Ingrese una letra:
u
Mensaje: NO ES UNA LETRA

Número de letras (encontradas,faltantes): (3,4)
Número de oportunidades restantes: 1
Estado Actual: _ a _ a _ _ a
Ingrese una letra:
q
Mensaje: NO ES UNA LETRA

Mensaje: Lo sentimos, no hay más oportunidades QQ

Excepciones

Ejercicio 9.

Crear una clase con un método main() que genere un objeto de la clase *Exception* dentro de un bloque *try*. Proporcione al constructor de *Exception* un argumento *String*. Para lanzar la excepción puede probar con cualquier caso. Capture la excepción dentro de una cláusula *catch* e imprima el argumento *String*. Añada una clausula *finally* e imprima un mensaje para demostrar que pasó por allí.

Ejercicio 10.

Inicializar un objeto a *null* de cualquier clase que ya haya definido en el Practico anterior. Trate de invocar un método a través de esta referencia. Ahora englobe el código con una clausula *try-catch* para probar la nueva excepción.

Ejercicio 11.

Escriba una clase donde Ud. pueda generar y capturar una excepción *ArrayIndexOutOf-BoundsException* (Índice de matriz fuera de límites).

Ejercicio 12.

Cree su propia clase de excepción utilizando la palabra clave *extends*. Escriba un constructor para dicha clase que tome un argumento *String* y lo almacene dentro del objeto como una referencia de tipo *String*. Escriba un método que muestre la cadena almacenada. Cree una cláusula *try-catch* para probar la nueva excepción.

Ejercicio 13.

Escribir un programa en Java que juegue con el usuario a adivinar un número. El ordenador debe generar un número aleatorio entre 1 y 500, y el usuario tiene que intentar adivinarlo. Para ello, cada vez que el usuario introduce un valor, el ordenador debe decirle al usuario si el número que tiene que adivinar es mayor o menor que el que ha introducido el usuario. Cuando consiga adivinarlo, debe indicárselo e imprimir en pantalla el número de veces que el usuario ha intentado adivinar el número. Si el usuario introduce algo que no es un número, debe indicarlo en pantalla, y contarlo como un intento.

Ejercicio 14.

Crea una clase Bombilla y un programa que cree objetos Bombilla y utilice todas las propiedades y métodos.

Propiedades:

1. estado: si está apagada o encendida (boolean). Por defecto false
2. potencia: en vatios (int)
3. color: color de la bombilla (String)
4. horas: horas que puede estar encendida (float). Por defecto 10

Métodos:

1. encender: si quedan horas enciende la bombilla, pone la propiedad estado = true y pide por teclado cuánto tiempo estará encendida. Resta el tiempo de la propiedad horas.
2. apagar: pone la propiedad estado = false
3. ver_estado: devuelve el estado de la bombilla
4. cambia_potencia: se le pasa una potencia por parámetro y la sustituye por lo que hay en la propiedad potencia
5. ver_potencia: devuelve la potencia de la bombilla
6. cambia_color: se le pasa el color por parámetro y si la bombilla está encendida la apaga y luego cambia el color.
7. ver_color: imprime por pantalla el color de la bombilla
8. recargar: pregunta cuántas horas quiere el usuario recargar y las suma a la propiedad horas

Clases Utiles

Ejercicio 15.

1. Crear una clase redondeo que permita ingresar un número float y una opción. La opción puede ser redondeo total, redondeo menor a 0,5 y mayor a 0.5. Utilizar la clase Math y el método de redondeo debe ser privado.
2. Pedir un número por teclado y preguntar si quiere ingresar más números. Posteriormente, mostrar la suma de los positivos y negativos, suma de los pares e impares, cantidad de positivos y negativos, cantidad de pares e impares y media de todos los números. Puede modelar a gusto del estudiante.
3. Realizar un programa que guarde datos de personas: nombre, edad y salario. Sin utilizar objetos. Luego pedir por teclado un nombre para buscar y mostrar edad y salario.
4. Implementar un programa que cree un array de dos dimensiones. Un método que pida números para llenar el array, otro que muestre ordenado por filas, otro que lo muestre ordenado por columnas, otro que busque el número mayor y lo muestre, y por último un método que pida un número y lo guarde en la fila y columna que diga el usuario.