PRÁCTICAS DEL CURSO DE JAVA

BÁSICOS:

1) Imprimir un mensaje de texto por la consola.
2) Definir dos variables enteras, asignarlas un valor y calcular la suma, después imprimir el resultado por la consola.
3) Leer un número desde teclado e imprimir un mensaje por pantalla si el número es par o impar.
4) Obtener los números primos del 1 al 100.

- Multiplicar dos números mediante sumas.
- Dividir dos números mediante restas.
- Calcular el factorial de un número.

ARRAYS:

5) Definir un vector inicializado con números enteros y contar el número de pares y el número de impares.
6) Calcular el número máximo y mínimo de un vector.
7) Calcular si los elementos de un array están ordenados ascendentemente.
8) Saber si los elementos de un array forman un número capicúa.

- Ordenación de arrays.
- Sumar dos arrays y guardar el resultado en un 3º.
- Buscar un elemento en un array.
- Hacer sumas parciales entre dos posiciones de un array.
- Contar el número de apariciones en un array.

String:

9) Hacer un programa que lea los argumentos de la línea de comando y luego los imprima al revés. Por ejemplo: java programa “hola” “adios”

Resultado:

aloh

soida

- Hacer un programa que a partir de una frase la parta en palabras. Se le pasará mediante los argumentos de main.
- Programa que cuente las vocales. Con args[].
- A partir de una frase generar 3 String en una metemos las vocales, en otra los números y en otra las consonantes.

OBJETOS:

10) Crear una clase que represente un punto en el plano, vendrá definido por sus dos coordenadas x e y. Tendrá que cumplir todas estas particularidades:
    a. Se podrá crear un punto a partir de otro punto, a partir de dos coordenadas o crearlo a 0.
    b. Métodos para sumar y restar puntos, pero no modificar el punto. La suma o diferencia se devolverá en un punto nuevo.
    c. Desplazar un punto a partir de un escalar.
    d. Calcular la distancia entre dos puntos: p_1 (x_1, y_1) y $p_2(x_2, y_2)$ $d = \sqrt{(x_1 - x_2)^2+(y_1 - y_2)^2}$
    e. Hacer un método que devuelva el punto como un String con este formato: (8.5, -9)
    f. Hacer otra clase a parte que defina dos o tres puntos y pruebe todos los métodos.

11) Implementar la clase Matriz, representará matrices y operaciones con matrices.
    a. Habrá dos 3 constructores, uno para crear en vacío, en otro los datos vendrán en un array de dos dimensiones, y por último le

indicaremos mediante dos enteros las dimensiones de la matriz. La clase también almacenará el número de filas y columnas.

b. Se implementará un método para multiplicar matrices.
c. El método toString pero saldrá así:

(5 6)
(2 1)

d. Métodos para extraer celdas, o asignar valores a celdas.
e. Y cuando se solicita un elemento deberá comprobar que está dentro de los límites.

12) Modelizar la clase coche. Un coche estará formado por las 5 ruedas y una radio. Cada rueda tendrá las siguientes propiedades: marca, modelo, presion, deAleaccion. También dispondrá de un método para inflar las ruedas. En el caso de la radio: Se podrá encender y apagar la radio, subir volumen.

HERENCIA:

13-0) A partir de la clase Punto2D crear la clase Punto3D con los mismos métodos y probarla.

13) Se pretende gestionar la información de los empleados de una empresa. Se encuentra dividida en cargos: jefes, empleados, y altos cargos que se compone de Directores y Directores Generales. La empresa dispone de 4 empleados, 2 jefes, 1 director y 1 director general. En total 8 personas. Como datos comunes a todos tenemos nombre, dni, cargo y sueldo base. Y las particularidades de cada cargo y el sueldo final se calcula:

- Empleados: dietas → sueldo base + dietas
- Jefes: comisiones, km, dias_libres (el precio del km y del dia_libre ya está preestablecido a un valor fijo) → Sueldo base + comisiones + km * precio + dias_libres * precio.

- Los altos cargos: valor del coche de empresa y objetivos. Y particularizando a Director: los viajes. Y Director General: dietas especiales.

Se pretende presentar un listado del nombre, dni, cargo y sueldo total por persona. Y además la suma total de todos los sueldos. Se hará un método distinto para cada cosa. Todas las variables de instancia se manejarán a través de métodos.

PAQUETES

14) Organizar la practica anterior en paquetes de la siguiente manera. Lo primero la clase empresa se parte en dos. Empresa que tendrá la definición y los métodos de la Empresa y la clase PruebaEmpresa que tendrá la función main. Se deberá crear la siguiente estructura de paquetes anidados. Cuando compilemos**: javac –d . *.java** nos tiene que crear la siguiente estructura de directorios. Y para ejecutar con:
java PruebaEmpresa

.
..
***.java**
PruebaEmpresa.class
<empresa> DIR
Empresa.class
<persona> DIR
Persona.class
<altoscargos> DIR
AltosCargos.class
Director.class
DirectorGeneral.class
<empleados> DIR
Jefe.class
Empleado.class

EXCEPCIONES:

15) Un programa que accederá a los argumentos del main. Primero hará una división de cualquier número entre el número de argumentos. Cuando no hayamos introducido argumentos se producirá un error. Y después intentará acceder a una posición no válida del array, por ej la 55. Capturar las dos excepciones y avisar al usuario.

16) Tenemos un programa que genera números aleatorios dentro de un bucle for de 1 a 32000, cada vuelta del bucle se generan dos números aletatorios y se resta. Y se realiza la siguiente operación: 12345 / resta, tenemos que capturar la posible división por cero. Los distintos cocientes de 12345 / resta se imprimen por pantalla y cuando salta la exception se contabiliza, una vez que termine el programa se imprime el número de divisiones por 0. Para generar números aleatorios:

```
import java.util.Random;
Random r = new Random();
  int b = r.nextInt();
```

17) Utilizando try anidados hacer una secuencia de programa que primero realice una división por cero, en el caso de que no hayamos metido argumentos en la función main. Usar al longitud del parámetros args[]. Después intentar acceder a una posición inexistente del array y volver a realizar una división por cero.

18) Trabajar sobre la clase empleado y crear un par de clases de Excepciones, una saltará cuando sueldo supere cierta cantidad y la otra cuando el cargo no sea EMPLEADO. Se hará un pequeño programa para probarlas. Quitar la herencia que tiene con persona y añadir los atributos que faltan. El control se hará en el constructor y en los métodos que tenemos para manejar las variables de instancia.

19) Se pretender tener un programa que calcule el área de varias figuras, en este caso queremos tener la clase triangulo y la clase rectángulo. Queremos tener un número ilimitado de rectángulos y triángulos cada uno se calcula de forma distinta. Queremos imprimir el resultado del área por pantalla.

StringTokenizer:

20) A partir de la clase Persona queremos cargar una serie de personas que las introducen por los argumentos del main. Cada persona lleva nombre, dni, sueldo y cargo. Cada campo va por dos puntos y cada persona por un ;
    a. Con este formato:
    pepe:61.989.907U:1000:administrativo;Juan:96.000.776J:1345:comercial;

Math:

21) Implementar una clase EcuacionGrado2 que resuelva ecuaciones de segundo grado.

Ax^2 + BX + C = 0 x = -B +- Raíz(B^2 – 4AC) / 2A

Después hacer la clase TestEcuacion que probará la clase.

Una vez que funcione implementar una clase RaizException que recogerá las excepciones producidas por una raíz negativa.

La clase EcuacionGrado2 lanzará esta Excepción cuando tenga que resolver una raíz negativa.

Properties:

22) Capturar las propiedades del sistema y imprimirlas por pantalla con un Iterator y una Enumeration. Usar HashTable.

Runtime

23) Hace un programa que abra el notepad, el programa java esperará hasta que se cierre el notepad y presentara el valor devuelto.

COLECCIONES:

24) A partir de un ArrayList implementar una clase Pila de enteros. Con los siguientes métodos. Constructor (si no se indica el número de elementos se creará con 10), métodos push, pop, para saber el tamaño de la pila, extraer elementos, saber si está llena o vacía, recorrer todos los elementos de la pila (con un iterator / bucle for :).

THREADS

25) Hacer un multihola, lanzaremos varios hilos que impriman un mensaje y que tengan un retardo para escribir. Implementar dos clases, una hereda de Thread y tendrá como parámetros un nombre y un retardo. Para que el retardo no sea igual para todos los hilos podemos usar el método random de Math. Podemos lanzar 25 hilos simultáneos.

26) Modificando los ejemplos del hilo padre (será el currentThread) y el hijo Hacer que el hijo de primero todos los mensajes y cuando termine arrancará el proceso padre e imprimirá sus mensajes.

27) Usando prioridades: Vamos a tener dos hilos (implementar una clase que implemente Runnable, tendrá un Thread, un contador y una prioridad). Desde main, damos máxima prioridad a currentThread y lanzamos a los dos hijos, uno con mas prioridad que otro. Dentro de la clase que implementa los hilos el método run contará cuantas vueltas podemos dar en un bucle while. Utilizar start() y stop(), desde el main llamaremos a start() de los dos hijos y el proceso padre se dormirá 10 sg, después espera a que terminen los hijos e imprime de cada hijo el número de vueltas de cada uno, el que tiene mas prioridad debería tener un número mas alto.

28) Ejercicio del productor y el consumidor.

a. Esquema del productor.
   i. Hereda de Thread.
   ii. Recibe la tubería a través de la cual con el método recoger() va a consumir 10 caracteres.
   iii. El carácter recogido lo imprime por pantalla.
   iv. Y se duerme un tiempo aleatorio, que no supere los 2 sg.

b. Esquema del consumidor
   i. Hereda de Thread y recibe la tubería.
   ii. A parte tiene otro atributo. Es el alfabeto.
   iii. String alfabeto = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   iv. Selecciona una letra al azar y la lanza a la tubería.
   v. char c = alfabeto.charAt((int)(Math.random()*26));
   vi. Imprime por pantalla el character lanzado,
   vii. Y se duerme un tiempo aleatorio no superior a 0,1 sg.

c. Esqueleto de la clase Tubería.

```
public class Tuberia {
    private char buffer[];
    private int siguiente;

     // Flags para saber el estado del buffer
    private boolean estaLlena;
    private boolean estaVacia;


    public Tuberia(){
        buffer = new char[6];
        siguiente = 0;
        estaLlena = false;
        estaVacia = true;
    }


    // Método para retirar letras del buffer
    public synchronized char recoger() {
        // No se puede consumir si el buffer está vacío

            // Se sale cuando estaVacia cambia a false

          // Decrementa la cuenta, ya que va a consumir una letra

            // Comprueba si se retiró la última letra, si es así //actualiza
            estaVacia

          // El buffer no puede estar lleno, porque acabamos de consumir

            // Avisa al productor.
```

```
            // Devuelve la letra al thread consumidor
        }


        // Método para añadir letras al buffer
        public synchronized void lanzar( char c ) {
            // Espera hasta que haya sitio para otra letra

            // Añade una letra en el primer lugar disponible

            // Cambia al siguiente lugar disponible (incrementa siguiente)

            // Comprueba si el buffer está lleno y actualiza estaLlena

           // No puede estar vacio el buffer porque acabamos de generar letra.

            // Avisa al consumidor
        }
}
```

d. Clase principal, crea la tubería y lanza los dos hilos.

ENTRADA / SALIDA

29) Mirar esta clase (File) en la API (paquete.io) crear una instancia de un fichero de nuestro disco y utilizando los métodos que sean necesarios, imprimir la siguiente info:
    a. Nombre del fichero
    b. Path
    c. Path absoluto
    d. Permisos de lectura y escritura
    e. Es fichero, es directorio
    f. Ultima modificación y tamaño del fichero.

30) A partir de un directorio listar todos los ficheros y directorios que haya dentro.

31) Hacer una modificación sobre el anterior para que cuando encuentre una carpeta liste el contenido de esta y así sucesivamente. Lo mas parecido al comando DOS: dir /s. Cada nivel de directorios se identara con un tabulación → \t. Al lado de cada directorio aparecerá <DIR>.

32) Hacer un programa para copiar ficheros. Implementar la clase copy que el constructor recibe el nombre del fichero a copiar y genera un fichero

llamado copia de… con el mismo contenido. La clase creada propagarán las excepciones hacia arriba.

SERIALIZACIÓN

33) Implementar una clase fecha compuesta por año, mes , dia y queremos hacer una prueba para serializar y deserializarla.

34) Se pretende gestionar una serie de facturas , que se puedan serializar y luego recuperar. Para la factura vamos a tener los siguientes campos:

a. Id
b. Fecha
c. Ordenante
d. Pagador
e. Importe

Se pide constructores y métodos para gestionar todas las propiedades de las facturas. Para el contenedor de facturas se implementarán métodos para insertar y recuperar facturas, con control si el contenedor de facturas está lleno o vacío.

A parte de estas 3 clases, se implementará otra clase para probar estas, que será a que crea el contenedor de facturas, insertará 3 facturas creadas con valores constantes y las serializará en un fichero, después hará la operación inversa que será recuperar los datos e imprimirlos por pantalla.