Programación II

Práctica 1

Diseño de Clases

Objetivos:

- Aprender los conceptos de clase, objetos, atributos, métodos.
- Aprender tipos de datos primitivos y String.

Ejercicio 1:

1- Un club tiene socios.

a) Crear la clase Socio con variables de instancia: nombre, número y variable de Clase: PróximoNúmero.
b) Agregar los métodos de acceso y modificación
c) Inicializar en 1 el próximo número.
d) Crear un socio, mostrar sus datos
e) Crear otro socio, mostrar sus datos

2- Crear la clase administrativo con atributos nombre y sueldo.
Incluir métodos de acceso y modificación. Agregar dos métodos de clase para prueba:
- ejemplo1: crea un administrativo con datos a ingresar por el operador y lo muestra
- ejemplo2: crea dos administrativos (usando el ejemplo1) e indica además cuál gana más.
Agregar teléfono en la clase administrativo. Hacer que el método toString lo incluya.

Ejercicio 2:

Cree una clase llamada Factura, que una ferretería podría utilizar para representar una factura para un artículo vendido en la tienda. Una Factura debe incluir un número de factura, un número de articulo, la descripción del articulo, la cantidad de artículos de ese tipo que se van a comprar y el precio por artículo.
Su clase debe tener un constructor que inicialice las variables de instancia. Proporcione los métodos get y set para cada variable de instancia. Además, proporcione un método llamado obtenerMontoFactura, que calcule el monto de la factura (es decir, que multiplique la cantidad por el precio por artículo) y después lo devuelva como un valor double. Si la cantidad no es positiva, debe establecerse en 0. Si el precio por artículo no es positivo, debe establecerse en 0.0. Escriba una aplicación de prueba llamada PruebaFactura, que demuestre las capacidades de la clase Factura.

Ejercicio 3:

Cree una clase llamada Fecha, que incluya tres variables de instancia: un mes (int), un día (int) y un año (int). Su clase debe tener un constructor que inicialice las tres variables de instancia, y debe verificar que los valores que se proporcionan son correctos. Proporcione los métodos get y set para cada variable de instancia. Proporcione un método mostrarFecha , para Imprimir la fecha en varios formatos, como

05/21/2010
Junio 14, 2012
20 Agosto 11
Usar constructores sobrecargados para crear objetos Fecha inicializados con fechas de los formatos solicitados para imprimir. En el primer caso, el constructor debe recibir tres valores enteros. En el segundo, debe recibir un objeto String y dos valores enteros. En el tercero debe recibir un valor entero, string y otro entero. Escriba una aplicación de prueba llamada PruebaFecha, que demuestre las capacidades de la clase Fecha.

Ejercicio 4:

El sector de ventas online de notebooks Lenovo paga a sus vendedores mediante comisiones. Los vendedores reciben $2000 por semana, más el 6% de sus ventas brutas durante esa semana. Por ejemplo, un vendedor que vende $50000 de mercancía en una semana, recibe $2000 más el 6% de $50000, un total de $5000.
Desarrolle una aplicación en JAVA que reciba como entrada los artículos vendidos por un vendedor durante cada día de la última semana, y que calcule y muestre los ingresos de ese vendedor. No hay límite en cuanto al número de artículos que un representante puede vender.

Ejercicio 5:

Crea una clase Punto que modele un punto en un espacio bidimensional. Tendrá dos atributos, x e y, que guardan las coordenadas. Habrá un constructor sin parámetros que crea un punto en (0, 0) y otro al que se le pueden pasar las coordenadas del punto. También habrá métodos para obtener las coordenadas y para imprimir el punto con el formato (x,y).

Ejercicio 6:

Crea una clase Rectángulo 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 7:

Una biblioteca maneja libros, algunos de ellos son originales y otros son fotocopias. No todos los libros se prestan.

a) Crear la clase Libro
b) Agregar atributos: título, original y prestable
c) Agregar métodos de instancia: ‘getOriginal’, ‘getTitulo’ y ‘getPrestable’.
d) Agregar métodos de instancia 'esOriginal' y 'sePresta' que retornen el valor booleano correspondiente.
e) Agregar métodos de instancia ‘setTítulo’ , ‘setOriginal’ y ‘setPrestable’
f) Crear un método main en una clase para prueba que permita obtener 2 instancias de Libro, uno de ellos es original y no se presta, el otro es fotocopia y se presta. Utilizar los métodos de

instancia para realizar estas operaciones. Mostrar los libros creados.

Ejercicio 8:
Definir una clase para manejar los artículos de un supermercado con los siguientes atributos: clave (int), descripción (string), precio (double), cantidad (int), y los siguientes métodos: constructor por defecto y constructor con parámetros y los métodos observadores y modificadores para cada atributo.

Ejercicio9:
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
- ♣ Métodos analizadores y modificadores. Para el número de cuenta no habrá modificador.
- ♣ 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.

Ejercicio 10:
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.
♣ analizadores y modificadores.
♣ 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.

Ejercicio 11:
Se requiere un programa que modele el concepto de un planeta del sistema solar. Un planeta tiene los siguientes atributos:

- Un nombre de tipo Sring con valor inicial de null.
- Cantidad de satélites de tipo int con valor inicial de cero.
- Masa en kilogramos de tipo double con valor inicial de cero.
- Volumen en kilómetros cúbicos de tipo double con valor inicial de cero.
- Diámetro en kilómetros de tipo int con valor inicial de cero.
- Distancia media al Sol en millones de kilómetros, de tipo int con valor inicial de cero.
- Tipo de planeta de acuerdo con su tamaño, de tipo enumerado con los siguientes valores posibles: GASEOSO, TERRESTRE y ENANO.
- Observable a simple vista, de tipo booleano con valor inicial false.

La clase debe incluir los siguientes métodos:

- La clase debe tener un constructor que inicialice los valores de sus respectivos atributos.
- Definir un método que imprima en pantalla los valores de los atributos de un planeta.
- Calcular la densidad un planeta, como el cociente entre su masa y su volumen.
- Determinar si un planeta del sistema solar se considera exterior.

Un planeta exterior está situado más allá del cinturón de asteroides. El cinturón de asteroides se encuentra entre 2.1 y 3.4 UA. Una unidad astronómica (UA) es la distancia entre la Tierra y el Sol= 149597870 Km.

- En un método main se deben crear dos planetas y mostrar los valores de sus atributos en pantalla.

Además, se debe imprimir la densidad de cada planeta y si el planeta es un planeta exterior del sistema solar.

Ejercicio 12:

1. Las Clases contienen principalmente _____ y _____ y los _____ son instancias de las mismas.
2. Queremos realizar el manejo de funcionarios de un supermercado, de los mismos se conoce su nombre y documento de identidad. Por lo tanto será necesario una Clase _____, con los atributos _____ y _____
3. Se tiene una Clase Avión con tres atributos: velocidadMaxima, nombreCompañia (nombre de la Compañía Aérea al que pertenece) y altitudMáxima. Los valores por defecto son:

velocidadMaxima = 1200km/h
nombreCompañia = “EmpresaXX”
altitudMaxima = 10000 m

Existe un constructor sin parámetros que inicializa los valores anteriores y otro con parámetros.
La clase posee los siguientes métodos:

setVelocidadMaxima(int unaVelocidadMaxima)
getVelocidadMaxima()
setNombreCompañia(String unNombreCompañia)
getNombreCompañia()
setAltitudMaxima(int unaAltitudMaxima)

getAltitudMaxima()

a) ¿ Cómo se crea un Objeto llamado "elAvion" de la Clase Avión?, Indique la(s) correcta(s):

a1) Avion elAvion = new Avion();
a2) elAvion Avion = new Avion();
a3) Avion elAvion = Avion();
a4) Avion elAvion = new Avion(1200,"EmpresaXX", 10000);

b) El objeto "elAvion" modifica su velocidad máxima a 1350 km/h y la altitud máxima a 15000m, escriba que métodos son necesarios invocar y cuales deben ser los parámetros requeridos.

c) Supongamos que se crean dos Objetos de la Clase Avión llamados "elAvion1" y "elAvion2".
Indicar qué imprime el siguiente código:

```
elAvion1.setVelocidadMaxima(1500);
elAvion2.setVelocidadMaxima(1400);
elAvion1.setVelocidadMaxima(1550);
System.out.println(elAvion1.getVelocidadMaxima() );
System.out.println(elAvion2.getVelocidadMaxima());
```

d) Supongamos que se crean tres nuevos objetos de la clase Avión denominados "elAvion3", "elAvion4" y "elAvion5" y luego se realizan las siguientes invocaciones:

```
elAvion3.setNombreCompañia("Iberia");
elAvion4.setNombreCompañia ("Pluna");
elAvion5.setNombreCompañia ("Iberia");
```

NOTA: Suponer la siguiente definición del método esDeLaMismaCompañia dentro de la Clase Avion:

```
public boolean esDeLaMismaCompañia(Avion unAvion) {
return (unAvion.getNombreCompañia().equals (this.getNombreCompañia());
}
```

Indicar que imprimen los siguientes códigos:

d1)
```
if (elAvion3.esDeLaMismaCompañia(elAvion4)){
    System.out.println("Son iguales");
}
else{
    System.out.println("Son diferentes");
}
```

d2)
```
if (elAvion3 == elAvion5){
    System.out.println("Son iguales");
}
else{
    System.out.println("Son diferentes");
}
```

d3)
```
if (elAvion3.esDeLaMismaCompañia(elAvion5){
    System.out.println("Son iguales");
}
else{
```

```
        System.out.println("Son diferentes");
}
```

Solución:

1) métodos, atributos, objetos
2) Funcionario, nombre, documento
3) a) a1 y a4; b) elAvion.setVelocidad(1350); elAvion.setAltitudMaxima(15000); c) 1550, 1400 d) d1) diferentes, d2) diferentes, d3) iguales

Programación II

Trabajo práctico *2*
Clases, Objetos y Colecciones

Objetivos:

- Repasar los conceptos de clase, objetos, atributos, métodos.
- Usar tipos de datos primitivos. Usar las clases String y ArrayList.
- Probar y depurar programas.
- Aprender distintas técnicas de prueba y depuración.
- Aprender a realizar un interfaz de usuario simple. Clase Scanner.

Ejercicio 1

Desarrollar una aplicación que nos permita almacenar información sobre películas (en DVD). La idea es crear un catálogo de todos los DVD que tenemos, hemos visto o deseamos tener o ver.

La funcionalidad que queremos que nos brinde el catálogo debe incluir como mínimo lo siguiente:

- Permitir ingresar información sobre los DVD.
- Permitir eliminar un DVD conociendo el título.
- Permitir modificar los valores de uno o más atributos conociendo el título.
- Listar:
  - todos los DVD
  - los DVD que tengo
  - los DVD que duran menos de un tiempo dado en minutos.
  - los DVD de un determinado director.
  - todos los DVD ordenados por título
- Informar:
  - la cantidad total de DVD.
  - la cantidad de DVD que tengo

Los detalles que queremos almacenar de cada DVD son:

- Título.
- Género.
- Tiempo de duración en minutos.
- Un atributo que indique si tenemos o no el DVD.
- Un comentario. (Excelente, Muy Buena, Buena, Regular, Mala)

Se pide:

a) Dibujar los diagramas de clase.
b) Programar la clase DVD e ir probando cada método.
c) Programar la clase Catálogo.
d) Realizar pruebas positivas y pruebas negativas.
e) Desarrollar pruebas automatizadas.
f) Documentar la aplicación usando javadoc.

Ejercicio 2

Agregar a la aplicación anterior una interfaz (de texto) de usuario.

Ejercicio 3

Desarrollar una aplicación similar la anterior pero que administre dos catálogos: uno de DVD y otro de CD. La funcionalidad del catalogo de CD es casi la misma que la de los DVD, excepto que queremos obtener listados por intérprete y también queremos saber cuantos temas tienen el CD.

Los detalles que queremos almacenar de cada CD son:

- Título del álbum.
- Intérprete (nombre de la banda o del cantante).
- Cantidad de temas que tiene el CD.
- Tiempo de duración en minutos.
- Un atributo que indique si tenemos o no el CD.
- Un comentario. (Excelente, Muy Buena, Buena, Regular, Mala).
- Género.

Ejercicio 4

Una vez que halla realizado la aplicación conteste:
Existe duplicación de código.

- V
- F

Si se modifica el campo duración de int a double para que pueda contener fracciones de tiempo debemos hacerlo una vez en la clase DVD y otra vez en la clase CD.

- V
- F

Esta aplicación sirve con **pequeñas modificaciones** si queremos que también administre un catálogo de libros.

- V
- F

Ejercicio 5:

Se desea diseñar un programa que registre libros (técnicos y novelas) para una librería y permita buscarlos, venderlos y verificar su stock.
Las novelas se clasifican como de ciencia ficción, romance, misterio, juveniles y policiales. Los libros técnicos se clasifican como de ingeniería, ciencias naturales o ciencias sociales.
Cada libro tiene un título, uno o más autores, una editorial, un año de edición y formato (tapas duras o edición económica). Los libros tienen además un código ISBN y capítulos, los que tratan una o más materias (en los técnicos) o es una simple división en las novelas.
La librería obtiene los libros por medio de proveedores que representan a una o más editoriales. De cada libro se tiene un stock (que puede ser cero). Al venderse un libro, el stock se actualiza. Si un cliente requiere un libro cuyo stock es cero, se puede realizar un encargo por parte del cliente. Esto significa que se pide el libro a un proveedor de la editorial del libro.

Ejercicio 6:

Crea una clase Banco que utilice una clase Cuenta, que tenga un nombre y una lista de cuentas bancarias. Definir en la clase los siguientes métodos:

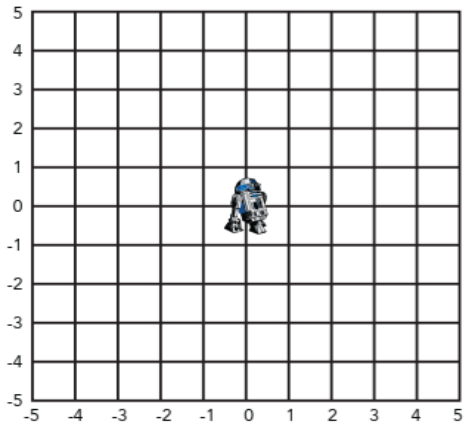
- ♣ Constructor donde se indique el nombre del banco.
- ♣ Método para consultar el nombre del banco.
- ♣ abrirCuenta(long dni, double saldo, double interes): crea un objeto Cuenta y lo guarda en la lista de cuentas. Devuelve un long que es el número de cuenta.
- ♣ cerrarCuenta(long numero): elimina la Cuenta con el número indicado. Devuelve true si se ha

realizado con éxito o false en caso de que no existiera.
♣ buscarCuenta(long numero): devuelve el objeto Cuenta con el número indicado.
♣ Método para imprimir el nombre del banco y la lista de las cuentas.

Ejercicio 7:
Se tiene un curso universitario el cual contiene una lista de estudiantes. Para cada estudiante se tienen los datos: nombre y apellidos del estudiante, código, número de semestre y nota final del estudiante. Se requiere implementar los siguientes métodos:

- Añadir un estudiante al curso: se ingresan por teclado los datos del estudiante. El código del estudiante debe ser único, si el código ya existe se debe generar el mensaje correspondiente.
- Buscar un estudiante de acuerdo con su código ingresado por teclado: si se encuentra muestra los datos del estudiante. De lo contrario, debe mostrar el mensaje correspondiente.
- Eliminar un estudiante de acuerdo con su código ingresado por teclado: si se encuentra muestra los datos del estudiante y se solicita una confirmación de la eliminación. Si no, debe mostrar el mensaje correspondiente.
- Calcular promedio del curso: sumar las notas de los estudiantes y dividirlas por la cantidad de estudiantes que tiene el curso.
- Obtener la cantidad de estudiantes que aprobó el curso: calcular el número de estudiantes que obtuvo un promedio mayor o igual a 4.0 y mostrarlo en pantalla. También se debe calcular el porcentaje de estudiantes que aprobó el curso.

Ejercicio 8:
Se requiere controlar un robot que existe dentro de un espacio bidimensional, como el que se muestra en la figura.



Este robot puede hacer giros de 90 grados en sentido contrario a las manecillas del reloj, uno a la vez, y avanzar hacia el frente una unidad en cada desplazamiento. Para lograr esto, se está considerando la clase Robot, cuyos atributos, constructores y métodos se describen a continuación.
Atributos:

• x, y, de tipo entero, que representan la posición horizontal y vertical, respectivamente, del robot dentro del espacio bidimensional.

• direccion, de tipo entero, para representar, en grados, la dirección que tiene el robot, que puede ser 0, 90, 180 o 270.

Constructores:

• Robot(), constructor que establece los valores por omisión 0, 0 y 0 para los atributos x, y y dirección, respectivamente.
• Robot(int xi, int yi, int dir), constructor que establece los valores en los parámetros xi, yi y dir para los atributos x, y y dirección, respectivamente.

Métodos adicionales a los getters y setters:

• girar(), que indica al robot que tiene que hacer un giro de 90 grados en sentido contrario a las manecillas del reloj.
• avanzar() hace que el robot avance una posición en la dirección que el robot tiene actualmente.

Implementa la clase Robot considerando los atributos, constructores y métodos descritos. Además, escribe una clase principal para crear un par de Robots y hacer que giren y avancen y, en cada movimiento, saber cuál es su ubicación y su dirección.

Programación II

Trabajo práctico 3
Herencia

Objetivos:

- Aprender el concepto de herencia.
- Comprender cómo la herencia fomenta la reutilización de software.
- Entender que son las superclases y las subclases.
- Comprender los distintos modificadores de acceso: privado, público y protegido.
- Utilizar miembros de superclases mediante super.

Ejercicio 1
Definir la clase Estudiante como subclase de Persona. El estudiante tiene además matricula.
Verificar que los métodos y variables de instancia definidos como protected se heredan.
Probar con private y public.

Ejercicio 2
Dibuje una jerarquía de herencia para las clases Cuadrilátero, Trapezoide, Paralelogramo, Rectángulo y Cuadrado.
Escriba esta jerarquía en Java. Los datos privados de Cuadrilátero deben ser los pares (x,y) para las cuatro esquinas. Escriba un programa para instanciar objetos de las distintas clase y que muestre el área de cada objeto (excepto Cuadrilátero).

Ejercicio 3
Supongamos que el departamento de sistemas de la empresa “Mi Luz S.A.” debe desarrollar un software para la liquidación de los haberes de sus empleados. La empresa cuenta con cuatro tipos de empleados:
- empleados asalariados
- empleados por horas
- empleados por comisión
- empleados base más comisión

Todo empleado tiene legajo y nombre y apellido. Un empleado asalariado tiene un sueldo mensual. Un empleado por horas tiene un sueldo básico, el número de horas trabajadas y el valor horario. Un empleado por comisión tiene una tasa de comisión y la cantidad e ventas en pesos. Un empleado por comisión tiene un sueldo base, una tasa y la cantidad de ventas.
Cada clase debe tener constructores apropiados, los métodos get y set. Escriba un programa que cree instancias de objetos de cada una de estas clases y que muestre toda la información asociada con cada objeto (incluyendo la información heredada).

Ejercicio 4
Modifique la aplicación que administra DVD y CD utilizando el concepto de herencia.

Programación II

Trabajo práctico 4
Herencia y Polimorfismo

Objetivos:

- Comprender el concepto de polimorfismo.
- Aprender a utilizar métodos sobreescritos para llevar a cabo el polimorfismos.

Ejercicio 1
Haga el seguimiento del siguiente programa y diga que imprime.

```
public class  redefinicion {
   public static void main(String[] args) {
      Madre m = new Madre();
      m.habla();
      Hija h = new Hija();
      h.habla();
      Nieta n = new Nieta();
      n.habla();
   }
}

class Madre {
   public void habla() {
     SystemIO.println("Soy de la clase Madre");
   }
}

class Hija extends Madre {
   public void habla() {
      SystemIO.println("Soy de la clase Hija");
   }
}

class Nieta extends Hija {
   public void otro() {
      SystemIO.println("Soy de la clase Nieta");
   }
}
```

Ejercicio 2
Haga el seguimiento del siguiente programa y diga que imprime.

```
public class conshered {
   public static void main(String[] args) {
      Hija h = new Hija(2);
   }
}

class Madre {
   public Madre(int mm) {
      SystemIO.println("Madre");
```

```
    }
}

class Hija extends Madre {
    public Hija(int ff) {
        super(ff);
        SystemIO.println("Hija");
    }
}
```

Ejercicio 3

Haga el seguimiento del siguiente programa y diga que imprime.

```
class Arte {
    Arte () {
        SystemIO.println("Constructor de Arte");
    }

    Arte (int i) {
        SystemIO.println("Constructor de Arte " + i);
    }
}

class Dibujo extends Arte{
    Dibujo () {
        super(100);
        SystemIO.println("Constructor Dibujo" );
    }
}

class DibujoAnimado extends Dibujo {
    DibujoAnimado () {
        SystemIO.println("Constructor Dibujo Animado" );
    }
}

public class pruebaArte {
    public static void main(String[] args) {
        DibujoAnimado donald = new DibujoAnimado();
    }
}
```

Ejercicio 4

Completar la siguiente tabla

Puede ser accedido desde	Un miembro declarado en una clase como			
	privado	*predeterminado*	*protegido*	*público*
Su misma clase				
Cualquier clase o subclase del mismo paquete				
Cualquier clase de otro paquete				
Cualquier subclase de otro paquete				

Ejercicio 5

Marque la respuesta correcta

a) _____ es una forma de reutilización de software, en la que nuevas clases adquieren los datos y comportamientos de clases existentes.

1. El encapsulamiento
2. La herencia
3. La composición

b) Los miembros _____ de una superclase pueden utilizarse solamente en la superclase o en las subclases.

1. private
2. protected
3. public

c) Los miembros _____ de una superclase son accesibles en cualquier parte donde el programa tenga una referencia a un objeto de esa superclase, o a un objeto de una de sus subclases.

1. private
2. protected
3. public

d) Los miembros de acceso protected de una superclase tienen un nivel de protección _____

1. mayor que los miembros de acceso private
2. entre los miembros de acceso private y los de acceso public
3. menor que los miembros de acceso public

e) Cuando se crea la instancia de un objeto de una subclase, el _____ de una superclase se llama en forma implícita o explícita.

1. constructor
2. modificador de las variables enteras
3. analizador de las variables principales

f) Los constructores de una subclase pueden llamar a los constructores de la superclase mediante la palabra clave _____.

1. super
2. boolean
3. switch

Conteste con verdadero o falso.

α) Los constructores de la superclase no son heredados por las subclases.
β) Una relación "tiene un" se implementa mediante la herencia.
χ) La clase Auto tiene una relación "es un" con su VolanteDireccion y sus Frenos.
δ) La herencia fomenta la reutilización de software comprobado, de alta calidad.
ε) Cuando una subclase redefine al método de una superclase utilizando la misma firma, se dice que la subclase sobrecarga a ese método de la superclase.

Ejercicio 6

Modifique la aplicación que administra DVD y CD utilizando el concepto de polimorfismo.

Ejercicio 7

La empresa "Never-NO" realiza transportes de diferentes tipos de carga, utilizando carreteras nacionales y provinciales. Para ello requiere vehículos especialmente acondicionados según la carga.

Cada transporte se identifica por el número de patente y se mantiene la siguiente información:
1. descripción de la carga
2. localidad de salida
3. localidad de llegada

Las localidades están codificadas
según el registro de códigos
postales vigente.

Datos adicionales según la carga a transportar:

Alimentos	Bultos	Ganado
Lista de tipos de alimentos, cantidad a transportar de cada uno y su peso.	Cantidad de bultos de cada tamaño	Cantidad de cabezas a transportar y peso promedio por cabeza

Para el caso de bultos, el tamaño de los mismos se clasifica según su peso:
Bulto MINI: 0.01 a 20 kg. Bulto MID: 21 a 100 kg. Bulto MAX: 101 a 150 kg.
Para el cálculo de peso se considera un promedio de dicho rango.

Escribir un programa aplicando el concepto de polimorfismo que muestre el costo del viaje para un vehículo dado, sabiendo que éste depende no sólo del trayecto a realizar sino de la carga a transportar.
Costo por peso transportado:

Hasta 1200 kg	$ 6.00 por kg.
Hasta 2400 kg.	$ 5.50 por kg.
Hasta 4000 kg.	$ 4.30 por kg.
Más de 4000 kg.	$ 3.60 por kg.

Los transportes se realizan entre 10 localidades. No siempre la tarifa de viajar de una localidad a otra es igual en caso de realizar el viaje inverso. Existe una matriz disponible de tarifas de transporte entre ciudades.

Escribir un método (y mostrar su uso) que, para dos transportes A y B, indique aquél de mayor costo.
Escribir un método en la clase aplicación que reciba un conjunto de transportes y muestre en pantalla su información según el tipo de transporte.

Programación II

Trabajo práctico 5
Clases abstractas y interfaces

Objetivos:

- Distinguir entre clases abstractas y concretas.
- Aprender a declarar métodos abstractos para crear clases abstractas.
- Comprender el concepto de interface.

Ejercicio 1

Reescriba el sistema de empleados de la empresa “Mi Luz S.A.” utilizando donde corresponda clases abstractas.

Ejercicio 2

Haga el seguimiento del siguiente programa y diga que imprime.

```
abstract class Poligono {
   public String color;
   public long altura;
   public long base;
   abstract long area();
   abstract void pintar(String color);
}

class Rectangulo extends Poligono {
   public Rectangulo( long x, long y) {
      altura = x;
      base = y;
}

   public void pintar( String color) {
      System.out.println("Rectangulo color " + color);
   }

   public long area() {
      return base * altura;
   }
}

class Cuadrado extends Poligono{
   public Cuadrado (long x) {
      base = x;
      altura = x;
}

   public void pintar( String color) {
      System.out.println("cuadrado color " + color);
   }
```

```
    public long area() {
        return base * altura;
    }
}

class  EjercicioPoligono {
    public static void main(String[] args) {
        Rectangulo rect = new Rectangulo(10, 5);
        pintarObjeto(rect, "Rojo");
        Cuadrado cuad = new Cuadrado(4);
        long i = areaObjeto(cuad);
        System.out.println("El area del cuadrado es " + i);
    }

    public static void pintarObjeto(Poligono p, String c) {
        p.pintar(c);
    }

    public static long areaObjeto(Poligono p) {
        return p.area();
    }

}
```

Ejercicio 3

Marque la respuesta correcta

g) El polimorfismo ayuda a eliminar la lógica de _____.

4. while
5. for
6. switch

h) Si una clase contiene al menos un método abstracto es una clase _____.

4. privada
5. abstracta
6. concreta

i) Las clases a partir de las cuales pueden instanciarse objetos se llaman _____..

4. privadas
5. abstractas
6. concretas

j) _____ implica el uso de una variable de superclase para invocar métodos en objetos de la superclase y subclase.

4. El polimorfismo
5. La herencia
6. El encapsulamiento

k) Los métodos abstractos se declaran utilizando la palabra clave _____.

4. interface
5. super
6. abstract

l) En una interfaz no se pueden declarar _____.

4. variables de instancia
5. métodos
6. constantes

Conteste con verdadero o falso.

ϕ) Todos los métodos de una clases abstracta deben ser declarados como métodos abstract.
γ) Una clase abstracta debe tener por lo menos un método abstracto.
η) Una clase se hace abstracta declarándola como abstract.
ι) Si una superclase declara un método como abstract, una subclase debe implementar a ese método para convertirse en clase concreta.
φ) Una interfaz es un conjunto de declaraciones de constantes y métodos abstractos.
κ) Una clase puede implementar una sola interfaz.
λ) Una interfaz puede ser implementada por muchas clases.

Ejercicio 4

Uno de los algoritmos más simples, pero menos eficiente para ordenar un conjunto de valores es el denominado SELECCIÓN.

a) Averigüe cómo funciona este algoritmo.
b) Prográmelo para ordenar un conjunto de números enteros.
c) Idem para un conjunto de números reales.
d) ¿Que diferencias hay entre b y c?
e) Considere ahora que deseamos ordenar un conjunto de palabras ¿cómo implementaría el método de selección en este caso?
f) Desarrollar una clase Avion con los atributos compañía (una única letra), número (un entero) y capacidad. Ahora queremos ordenar los aviones por compañía y para una misma compañía por número. Implemente el método de selección para este caso. ¿Qué diferencias hay con e?
g) Implemente el método de selección de manera que ordene cualquier tipo de objetos. Aplíquelo para ordenar los aviones.
h) Desarrollar una clase Libro con los atributos título y autor y que implemente la interfaz Comparable de tal manera que permita ordenar por autor y para un mismo autor por título.
i) Supongamos ahora que queremos tener dos ordenamientos de Libros, uno por autor y titulo y otro por titulo y autor. ¿Cómo haría?

Ejercicio 5

Agregar al proyecto de DVD y CD listados con distintos ordenamientos. Por ejemplo por género, por duración, etc. Se debe usar la versión del algoritmo de selección desarrollada en g.

Programación II

Trabajo práctico 6
Interfaz gráfica de usuario

Objetivos:

- Comprender los principios de diseño de interfaces gráficas de usuario (GUI).
- Crear interfaces gráficas de usuario.
- Aprender acerca de los paquetes que contienen componentes relacionados con las GUIs, class e interfaces para el manejo de eventos.
- Crear y manipular botones, etiquetas, listas, cajas de texto, paneles, etc.

Ejercicio 1
Agregar al proyecto una interfaz gráfica de usuario.

Programación II

Trabajo práctico 7
Excepciones

Objetivos:

- Comprender el manejo de excepciones y errores.
- Utilizar try, throw y catch para detectar, indicar y manejar excepciones.
- Comprender la jerarquía de excepciones de Java.
- Declarar nuevas excepciones.
- Aprender el procesamiento simple de archivos.

Ejercicio 1

Haga el seguimiento del siguiente programa y diga que imprime.

```
public class UsoExcepciones {
   public static void main(String[] args)    {
      try{
        lanzarExcepcion();
       }
      catch (Exception excepcion) {
        System.out.println("La excepcion se manejo en el main");
      }
        noLanzaExcepcion();
}

public static void lanzarExcepcion() throws Exception {
   try {
      System.out.println("El metodo lanzarException");
      throw new Exception();
   }
   catch (Exception excepcion) {
      System.out.println("La excepcion se manejo en el metodo
                                    lanzarExcepcion");
   }
}

public static void noLanzaExcepcion(){
    try  {
      System.out.println("El metodo noLanzaExacpcion");
    }
    catch (Exception e)     {
      System.out.println(e);
    }
}
}
```

Ejercicio 2

Haga el seguimiento del siguiente programa y diga que imprime.

```
public class UsoExcepciones1 {
     public static void main(String[] args)    {
        try{
           lanzarExcepcion();
        }
        catch (Exception excepcion) {
           System.out.println("La excepcion se manejo en el main");
       }
         noLanzaExcepcion();
     }

public static void lanzarExcepcion() throws Exception {
     System.out.println("El metodo lanzarExcepcion");
     throw new Exception();
}

public static void noLanzaExcepcion(){
     try  {
        System.out.println("El metodo noLanzaExecpcion");
     }
     catch (Exception e)      {
       System.out.println(e);
     }
 }

}
```

Ejercicio 3

Marque la respuesta correcta

a) Uno de los siguientes no es un ejemplo de excepción.

1. División por cero
2. Índice de un arreglo fuera de rango
3. Ciclo infinito

b) Una excepción se atrapa con la estructura:

1. if else if
2. switch
3. try catch

c) El programador encierra en un bloque _____ el código que puede generar una excepción.

1. catch
2. try
3. exception

d) El bloque try va seguido por _____ o más cláusulas catch.

1. cero
2. una
3. dos

Conteste con verdadero o falso.

a) La clase Exception hereda de la clase Error
b) El espíritu detrás del manejo de excepciones es permitir a los programas atrapar y manejar los errores, en vez de dejar que ocurran y sufrir las consecuencias.
c) Cuando un método lanza una excepción, ésta debe ser atrapada en el mismo método
d) Una excepción puede ser propagada “hacia arriba”.
e) El código encerrado en un bloque catch no puede ser vacío.

Ejercicio 4

Agregar el manejo de excepciones al proyecto de DVD y CD. Por ejemplo se desea listar por género y no hay ninguna película del mismo.
Además debe almacenar la información de manera permanente, de tal modo que pueda ser usada más adelante.

Ejercicio 5:

Se requiere desarrollar un programa para verificar si una contraseña es válida. Los requisitos de la contraseña son los siguientes:

- Mínimo 8 caracteres.
- No debe tener espacios en blanco.
- Debe tener por lo menos un carácter, un carácter en mayúscula, un número y un carácter especial.
- La contraseña se debe ingresar dos veces para su confirmación.

Se lanzarán excepciones si no se cumplen dichos requerimientos y si las dos contraseñas no son iguales.

Programación II

Trabajo práctico 7
Excepciones y Archivos

Objetivos:

- Aprender a crear y usar archivos texto.

Ejercicio 1

Escriba un programa que reciba una de estas opciones o las dos:

c	para contar caracteres
l	para contar líneas

y un nombre de archivo sobre el cual realizar la operación indicada.

Ejercicio 2

Construya un programa que cuente la cantidad de ocurrencias de un dado carácter en un archivo de texto.

Ejercicio 3

Construya un programa que convierta un archivo de texto a mayúsculas o minúsculas (según se especifique) y lo copie en otro archivo.

Ejercicio 4

Se tiene un archivo de texto en el que cada línea contiene la siguiente información:

NOMBRE DEL PAÍS (máximo 30 caracteres)
POBLACIÓN (real, en millones de habitantes).
PBI (real)

Escriba un programa que genere 2 archivos de texto, uno con los países con población menor a 30 millones de habitantes y el otro con los restantes.

Ejercicio 5

Haga una nueva versión del programa anterior que reciba la cantidad de habitantes a tener en cuenta para la generación de archivos. Además, si se recibe como opción:

PBI	sólo grabar PAÍS y PBI.
POB	sólo grabar PAÍS y POBLACIÓN.

Si no se recibe opción, grabar todos los datos.

Ejercicio 6

Definiremos un archivo que contendrá datos bancarios de una serie de clientes, al que llamaremos Clientes. Para cada uno de los siguientes puntos define un método y ejecútalo en el main para testearlo.

1) Define un ArrayList que contenga los nombres y apellidos de 10 clientes. Graba este array completo en el archivo Clientes e imprímelo en la pantalla.

2) Ingresa luego por teclado 5 datos más. Deben quedar luego los 15 datos grabados en el archivo, donde los 5 datos recién ingresados estén al principio del archivo. Imprímelo en pantalla
3) Define ahora un archivo llamado Clientes2 donde además aparezca el número de cuenta de cada cliente, su nombre y apellido y el tipo de cuenta. Los tipos de cuentas posibles son Caja de ahorro o Cuenta corriente.

Ejercicio 7

Hacer un programa que permita actualizar un archivo de datos a elección (puede ser un archivo de libros, de temas musicales, de películas, de autos, de estudiantes, etc). También se debe poder realizar consultas e informes.

Realizar un menú similar al siguiente:

- Agregar un nuevo elemento (alta).
- Eliminar un elemento (baja lógica).
- Modificar los datos de un elemento existente.
- Consultas
  - Dado un valor de la clave mostrar los datos del elemento correspondiente.
  - Cantidad de elementos.
  - Cantidad de elementos que cumplen con alguna condición.
- Informes
  - Listado de todos los elementos.
  - Listado de los elementos que cumplen alguna condición.
- Salir

Ejercicio 8

Se requiere implementar una clase vendedor que posee los siguientes atributos: nombre (tipo String), apellidos (tipo String) y edad (tipo int).
La clase contiene un constructor para inicializar los atributos de la clase. Además, la clase posee los siguientes métodos:

- Imprimir: muestra por pantalla los valores de sus atributos.
- Verificar edad: este método recibe como parámetro un valor entero que representa la edad del vendedor. Para que un vendedor pueda desempeñar sus labores se requiere que sea mayor de edad (mayor de 18 años). Si esta condición no se cumple, se lanza una excepción de tipo IllegalArgumentException con el mensaje “El vendedor debe ser mayor de 18 años”. Además, se evalúa si la edad se encuentra en el rango de 0 a 120, si no se cumple, se genera una excepción de tipo IllegalArgumentException con el mensaje “La edad no puede ser negativa ni mayor a 120”. Si la edad cumple estos requerimientos se pueden instanciar el objeto vendedor.

Ejercicio 9:

Un equipo de programadores desea participar en una maratón de programación. El equipo tiene los siguientes atributos:

- Nombre del equipo (tipo String).
- Universidad que está representando el equipo (tipo String).
- Lenguaje de programación que va a utilizar el equipo en la competencia (tipo String).
- Tamaño del equipo (tipo int).

Se requiere un constructor que inicialice los atributos del equipo. El equipo está conformado por varios programadores, mínimo dos y máximo tres. Cada programador posee nombre y apellidos (de tipo String). Se requieren además los siguientes métodos:

- Un método para determinar si el equipo está completo.
- Un método para añadir programadores al equipo. Si el equipo está lleno se debe imprimir la excepción correspondiente.
- Un método para validar los atributos nombre y apellidos de un programador para que reciban datos que sean solo texto. Si se reciben datos numéricos se debe generar la excepción correspondiente. Además, no se permiten que los campos String tengan una longitud igual o superior a 20 caracteres.
- En un método main se debe crear un equipo solicitando sus datos por teclado y se validan los nombres y apellidos de los programadores.

Ejercicio 10

Se requiere un programa que almacene y recupere información almacenada en un archivo sobre una asignatura universitaria. La información de la asignatura comprende:

- Código de la asignatura (tipo int).
- Nombre de la asignatura (tipo String).
- Cantidad de créditos de la asignatura (tipo int).

La clase asignatura debe tener un constructor que inicialice sus atributos y los siguientes métodos:

- imprimir: muestra en pantalla los valores de los atributos de una asignatura.
- leerAsignatura: lee los contenidos de un archivo que tiene almacenada la información de una asignatura y los muestra en pantalla.
- escribirAsignatura: escribe los contenidos de un objeto asignatura en un archivo.

Se debe desarrollar una clase de prueba que cree una asignatura, la almacene en un archivo y, luego, la recupere y muestre su información en pantalla.

Ejercicio 11

Una inmobiliaria posee un listado de inmuebles para vender y arrendar. Cada inmueble tiene como atributos: dirección, ciudad, tipo (de arriendo o venta). Se requiere que un programa puede agregar, editar y eliminar inmuebles. Además, el listado de inmuebles junto con sus datos se debe guardar en un archivo.

Programación II

Trabajo práctico 8
Dase de Datos

Objetivos:

Aprender a usar base de datos

Ejercicio 1:
Crear en MySQL una base de datos que contenga una tabla Clientes con legajo, nombre, apellido; otra tabla Productos con código, descripción y precio unitario y una tercera tabla con Ventas, con número de venta, número de legajo de cliente, código de producto y cantidad.
Cargar las tablas y mostrar utilizando consultas desde un programa, los resultados de las siguientes consultas:

a) Listado de ventas en el que figuran: Apellido y Nombre del cliente, código de producto, cantidad vendida, total (precio unitario * cantidad), ordenado por Apellido del cliente.

b) Suma de ventas realizadas (sumatoria de cantidades vendidas).

c) Clientes que aparecen en la tabla de ventas realizadas, con todos sus datos, sin repetir y cuantas compras realizó.

Ejercicio 2:
Defina una aplicación de consulta completa para la base de datos Libros. La cual está compuesta por la tabla Autores (IdAutor, Apellido, Nombre), la tabla Titulo (ISBN, Titulo, Edición, Anio), la tabla ISBNAutor que relaciona las dos tablas anteriores (Id, IdAutor, ISBN).

Realiza las siguientes consultas:
a) seleccionar todos los autores de la tabla autores.

b) seleccionar un autor específico y mostrar todos los libros de ese autor. Incluir el título, año e ISBN. Ordenar la información alfabéticamente, en base al apellido y nombre de pila del autor.

c) Seleccionar una editorial específica y mostrar todos los libros publicados por esa editorial. Incluir el título, año e ISBN. Ordenar la información alfabéticamente por título.

Ejercicio 3:
Siguiendo con el ejercicio anterior agregar las siguientes acciones a la aplicación:
a) agregar un nuevo autor.
b) editar la información existente para un autor.
c) agregar un nuevo título para un autor.

Ejercicio 4:
Una agencia de publicidad necesita una base de datos para registrar todas sus campañas en la web. Sus clientes tienen un nombre, una dirección postal, el número de teléfono y una dirección de email.
Cada cliente puede contratar varios anuncios. Los anuncios quedan identificados por un código y se caracterizan por un nombre, tipo (banner, popup, enlace patrocinado,...), título, contenido, categoría (tipo del producto que anuncia) y precio. Los anuncios pueden aparecer en más de una página web.
Cada web se caracteriza por su URL, nombre y tópico de interés. También se debe almacenar la fecha de inicio y de fin de la aparición del anuncio en la página web.
Desarrolle una aplicación de tipo AMBC (Alta-Baja-Modificación-Consulta) que administra la base de datos.

Ejercicio 5:
Una empresa de alquiler de vehículos desea conocer en todo momento el estado de su flota. La empresa tiene diversas oficinas repartidas pro todo el territorio argentino. Cada oficina se identifica por un código único y se caracteriza por la ciudad en la que se encuentra y su dirección completa (clalle, número y código postal) y teléfono. En cada oficina hay disponible un conjunto de autos, de los cuales se conoce su matrícula, el grupo al que pertenece: A, B, C, D, E (depende del tipo y tamaño del vehículo), la marca, el modelo, el número de puertas, el número de plazas, la capacidad del maletero y la edad mínima exigida para el alquiler. Para llevar el control del estado de cada vehículo, la empresa mantiene un registro de todos lod alquileres que ha sufrido, indicando para cada uno de ellos el nombre del conductor, su DNI, su dirección, un teléfono de contacto y un número de tarjeta de crédito sobre la que realizar los cargos correspondientes. Además de esta información de los clientes, para cada alquiler se almacena su duración (en días), el tipo de seguro contratado y el precio total.
Desarrolle una aplicación del tipo ABMC que permita administrar la base de datos.

Ejercicio 6:
Una pequeña floristería desea ampliar su negocio y realizar ventas a través de Internet. Y para ello necesita crear una base de datos.
Cada pedido incluye un número de pedido, la fecha de venta, el importe total y una lista con las flores solicitadas y en qué cantidad. Las flores se identifican mediante un código, su nombre y el precio de venta.Las flores pertenecen a una especie determinada. Para cada especie, se almacena el nombre, la época de floración, la estación de plantación, el tipo de suelo apropiado y el tiempo de exposición recomendado.
Desarrolle una aplicación del tipo ABMC que permita administrar la base de datos.

Ejercicio 7:
Prestame.com es una empresa que se dedica al préstamo entre personas. Los prestadores que desean prestar dinero a otros se registran con un id, su nombre y la cantidad de dinero disponible para las operacioens. Los deudores se identifican por su id y además el sistema almacena su nombre y un valor de riesgo en función de su situación personal.
Cuando el deudor solicita un préstamo, se añade un código de préstamo, el importe toal, el plazo de devolución, su interés y la finalidad del mismo. Los prestadores indican qué cantidad quieren aportar a un préstamo. Un prestador puede aportar distintas cantidades parciales a varios préstamos.

Ejercicio 8:
Una empresa aérea necesita una base de datos para registrar la información de sus vuelos.
Los vuelos tienen un identificador único. Además, cada vuelo tiene asignado un aeropuerto de origen y no de destino (se asume que no hay escalas). Los aeropuertos están identificados por unas siglas únicas (ejemplo: EZE-BsAs, BCN-Barcelona, MIA- Miami). Además, de cada aeropuerto se guarda el nombre de la ciudad en la que está situado y el país. Cada vuelo es realizado por un avión. Los aviones tienen una matrícula que los identifica, el fabricante, un modelo e información sobre su capacidad (número máximo de pasajeros) y autonomía de vuelo (en horas). La asignación de aviones a vuelos no es única, así que es necesario saber la fecha en la que un avión realizó cada uno de los vuelos asignados.

Ejercicio 9:
Partiendo de la base de datos, la cual contiene un log de sensores de temperatura, humedad y radiacion UV. Escribir las consultas SQL que permitan listar la siguiente informacion:
1) Listar las últimas 3 mediciones de temperatura con las columnas “valueSensor”,”unit” y “timestamp”.
2) Listar las últimas 2 mediciones de temperatura del sensor del equipo con las columnas “valueSensor”,”unit” y “timestamp”.

3) Listar los momentos del dia en que la humedad fue mayor al 50%, indicando la columna de “timestamp”. Mostrar en formato fecha y hora (utilizar la funcion “datetime(xxxx,’unixepoch’)”)
4) Listar la lista de sensores disponibles en el sistema indicando sus nombres, unidad que miden y descripcion del tipo de sensor.
5) Listar la lista de sensores de temperatura disponibles en el sistema indicando sus nombres, unidad que miden y descripcion del tipo de sensor.
6) Consultar el valor de radiacion UV medido promedio filtrando por tipo de sensor, no por id de sensor. (Utilizar la funcion “avg()”)
7) Consultar el valor de temperatura máxima registrado indicando las columnas de valor, unidad y timestamp. (Utilizar la funcion “max()”). Filtrar por tipo de sensor.

Programación II

Trabajo práctico B-1
Elementos básicos de la programación en Lenguaje C

Objetivos:

Introducción al lenguaje C. El alumno aprenderá a usar el editor y el compilador de Lenguaje C, además de repasar los conceptos básicos de la programación vistos en Programación I. Se deberá diseñar y codificar un programa en C para cada enunciado.

Ejercicio 1

Escribir un programa que muestre un aviso de bienvenida en la pantalla.

Ejercicio 2

Escribir un programa que calcule la distancia de un maratón en km, sabiendo que un maratón tiene 26 millas y 385 yardas. Una milla tiene 1760 yardas.

Ejercicio 3

Hacer un programa que sume los primeros 10 números naturales.

Ejercicio 4

Escribir un programa para resolver una ecuación de segundo grado teniendo en cuenta que la ecuación puede tener raíces complejas.

Ejercicio 5

Escribir un programa en C que escriba una tabla de dos columnas para la conversión entre las temperaturas en grados Fahrenheit −comprendidas entre 0 ºF y 300 ºF, según incrementos de 20 ºF− y su equivalente en grados centígrados. Se realizarán dos versiones de este programa: una llamada **temp1.c** que empleará un bucle **while**. La otra versión se llamará **temp2.c** y utilizará un bucle **for**.

La conversión entre grados Centígrados y grados Fahrenheit obedece a la fórmula:

$$ºC = 5 * (ºF – 32)/9$$

Programación II

Trabajo práctico B-2

Procesamiento de datos almacenados en arreglos y uso de ciclos.

Objetivos:

El alumno aprenderá a hacer operaciones repetitivas para procesar secuencias de datos almacenadas en arreglos usando el ciclo for.

Ejercicio 1

Encontrar el máximo elemento de un conjunto de números de tamaño n. El valor n se deberá ingresar desde teclado.

Ejercicio 2

Ordenar un conjunto de números enteros por el método de Burbujeo.

Ejercicio 3

Supongamos que concurren 3 partidos a las elecciones y que la provincia o distrito electoral dispone de 2 escaños. El primer escaño se lo llevará el partido más votado. Para el segundo escaño se dividen los votos de cada partido entre el número de escaños obtenidos más uno (el partido que no tenga todavía ningún escaño se dividirá entre 1) y se calculan los restos de dichas divisiones. El escaño se asigna al partido que tras esta operación tenga un resto más elevado.

Ejercicio 4

Hacer un programa que multiplique una matriz por un vector. Tener en cuenta los casos en los que no se puede realizar la operación.

Ejercicio 5

Hacer un programa que multiplique dos matrices. Tener en cuenta los casos en los que no se puede realizar la operación.

Ejercicio 6

Hacer un programa para determinar si un número es primo.

Ejercicio 7

Calcular el determinante una matriz de 3x3.

Ejercicio 8

Habrás oído hablar de la historia de un poderoso sultán que deseaba recompensar a un estudiante que le había prestado un gran servicio. Cuando el sultán le preguntó qué recompensa deseaba, éste le señaló un tablero de ajedrez y solicitó simplemente 1 grano de trigo por la primera casilla, 2 por la segunda, 4 por la tercera, 8 por la siguiente, y así sucesivamente. El sultán, que no debía andar muy fuerte en matemáticas, quedó sorprendido por la modesta de la petición, porque estaba dispuesto a otorgarle riquezas muy superiores: al menos, eso pensaba él. Hacer un programa para calcular cual fue la recompensa.

Programación II

Trabajo práctico B-3
Funciones

Objetivos:

El alumno apredenderá a identificar las diferentes tareas que forman un programa y realizar funciones para llevarlas a cabo. De esta forma aprenderá a hacer funciones reusables y a hacer los programas más eficientes escribiendo menos código y reusando funciones existentes. También aprenderá a usar la herramienta de la recursividad y a aplicarla correctamente.

Ejercicio 1

Escribir un programa que deberá realizar las siguientes operaciones:

1. Preguntar al usuario con cuántos números desea trabajar.
2. Leer y almacenar los números convenientemente.
3. Hallar su media aritmética utilizando para ello una función a la que llamarás ***media( )***.
4. Hallar el máximo y el mínimo llamando a sendas funciones ***maximo( )*** y ***minimo( )***, cuya programación realizaste en la práctica anterior.
5. Hallar la semisuma de los valores máximo y mínimo mediante la función ***semisuma( )***.
6. Mostrar por pantalla los valores obtenidos: media, máximo, mínimo y semisuma.
7. Tanto el programa principal como las funciones deberán estar en el mismo fichero ***numeros.c***.

Ejercicio 2

Hacer una función para calcular la norma infinito de una matriz:

$$\|A\|_\infty = \max_i \sum_j |a_{ij}|$$

Ejercicio 3

Hacer una función para calcular el valor de exp(x) usando su desarrollo en serie de Taylor a n términos.

$$exp(x) = 1+x/1! + x^2/2! + x^3/3! + \ldots$$

Ejercicio 4

Hacer funciones para calcular:

a) cantidad de dígitos de un número entero
b) sumar los dígitos de un número entero
c) obtener el dígito de mayor valor de un número entero
d) invertir los dígitos de un número entero obteniendo un número nuevo
e) obtener el k-ésimo dígito de un número entero contando desde la derecha hacia la izquierda, ejemplo: 12345 si k = 2 el dgito es 4 y si k = 5 el dígito es 1.

Ejercicio 6

Realizar funciones recursivas para realizar el ej. 4 y 5.

Ejercicio 7

α) Dado un número de un dígito entre 1 y 9, escribir una función recursiva que devuelva el número dado como una secuencia de unos. Ejemplo:

Número: 4 Respuesta: 1111

β) Escribir una función recursiva que cuente cuántas veces se repite un determinado dígito en un número. Ejemplos:

Número: 23252 Dígito: 2 Respuesta: 3

Número: 23252 Dígito: 4 Respuesta: 0

χ) Escribir una función recursiva que dado un número obtenga otro con los dígitos permutados. Ejemplo:

Número: 531 Respuesta: 135

δ) Dado un número entero positivo, escribir una función recursiva que devuelva la representación binaria del mismo. Ejemplo:

Número: 4 Respuesta: 100

ε) Dado un número en su codificación binaria, hacer una función recursiva que halle su representación decimal. Ejemplo:

Número: 10101 Respuesta: 21

ϕ) Escribir una función recursiva para hallar el máximo común divisor entre dos números enteros positivos. Ejemplo:

a = 180; b = 75 Respuesta: 15

Nota: Investigar cómo funciona el algoritmo de Euclides para hallar el máximo común divisor.

γ) Escribir una función recursiva para descomponer un número es sus factores primos. Ejemplo:

Número = 180 Respuesta: 2, 2, 3, 3, 5

η) Escribir una función recursiva para hallar la suma de la serie:

*1*2 + 3*4 + 5*6 + . . . + (2n-1)(2n)*

ι) Escribir una función recursiva que reciba un número entero n y despliegue la secuencia: n, n-1, n-2, . . .,1.

Ejercicio 8

Hacer funciones recursivas para:

♋ encontrar la suma de los elementos de un vector de n elementos

♌ encontrar el elemento de mínimo valor de un vector de n elementos

Ejercicio 9

Hacer una función recursiva para calcular el determinante de una matriz de nxn elementos.

Programación II

Trabajo práctico B-4
Punteros

Objetivos:

El alumno aprenderá a usar el acceso directo a memoria para manipular datos en los programas.

Ejercicio 1

Dadas las siguientes definiciones y asignaciones:

```
int a[10] = {1,5,8,4,9,11,12,7,6,5};
int *p, *q;
int i;

p = &a[0];
q = &a[1];
```

a) **Grafique** como serían las asignaciones de memoria a las variables e indique con una flecha las posiciones a las que apuntarían las variables punteros.

b) **Indique gráficamente** el estado de la memoria luego de ejecutar cada una de las siguientes sentencias en forma secuencial:

```
1. i = *p + *q;
2. *p += *q; /* o lo que es lo mismo: *p = *p + *q; */
3. *(p+1) = i+10;
4. i = *(q-1) / *(p+9);
5. *p = *(p+10-3);
6. q = p+5;
```

c) ¿Qué diferencia hay entre el nombre de un arreglo y un puntero?

d) ¿Cómo sabe el compilador el tamaño de un objeto al que apunta un puntero?

Ejercicio 2

Dado el siguiente programa.

```
# include <stdio.h>

    int main ( )
    {
    int arr[10], i;
    int *p_arr;

    for (i = 0; i < 10; i++)
        arr[i] = i;
    p_arr = &arr[0];
    for (i = 0; i < 10; i++) {
        printf ("%d -- %d \n", *p_arr, *p_arr * *p_arr);
    p_arr++;
    }
    return (0);
    }
```

a) **Ejecute** paso a paso el código e indique cuál es la salida del programa.
b) ¿Es **necesario** colocar paréntesis en el cálculo de la multiplicación? Justifique.
c) Indique otras formas de escribir **sintácticamente correcto** el printf del programa.

Ejercicio 3

Dado el siguiente programa:

```
# include <stdio.h>
int main ( ) {
char palabras[7];
char *p_arr;
char cartel[20]= {'L','o','s',' ','d','a','t','o','s',' ','s','o','n',':'};
printf("%s \n", cartel);
printf("%s \n", palabras);
for (i = 0; i <= 7; i++)
palabras[i] = 'z'-i;
printf("%s \n", palabras);
*(palabras + 3) = '\0';
printf("%s \n", palabras);
getchar();
p_arr= &palabras[7];
for (i = 7; i >0 ; i++) {
printf ("Posicion:%d - %c \n",i-1, *(p_arr-1));
p_arr--;
}
getchar();
return (0);
}
```

a) ¿Es ejecutable? En caso de no serlo corregirlo para que muestre los caracteres en orden inverso al ingresado.
b) Explicar qué sucede si se olvida el carácter nulo como último carácter de una cadena (string). Justifique la salida de cada printf

Ejercicio 4

a) Hacer una función para invertir una cadena de caracteres.
b) Escribir una **función** que reemplace en la cadena de caracteres denominada "cad", todas las apariciones del carácter llamado "viejo" por el carácter indicado en "nuevo".

Ejercicio 5

Realizar un programa que lea 25 números enteros positivos, determine el mayor número par y el menor número impar de todos. Se deben utilizar funciones que calculen el menor, el mayor y la determinación de si es par o no. Las funciones definidas deberán retornar el resultado correspondiente. El programa principal debe imprimir los 25 números y lo retornado por las funciones.

Ejercicio 6

Escribir un programa que lea dos cadenas de caracteres alfabéticos y un número. El programa debe insertar la segunda cadena en la primera, a partir del carácter de la primera cadena que está en la posición indicada por el número. Luego deberá imprimir la primera cadena.

Programación II

Trabajo práctico B-5
Estructuras

Objetivos:

El alumno aprenderá a crear sus propios tipos de datos usando estructuras que permiten la combinación de elementos de distintos tipos.

Ejercicio 1

Diseñar una estructura para almacenar un número complejo. Implementar funciones para realizar la suma, resta, producto y división de números complejos.

Ejercicio 2

Diseñar una estructura para almacenar las coordenadas de un punto en el plano. Implementar funciones para realizar las siguientes operaciones: trasladar el punto, hallar la distancia del punto al origen, hallar la distancia entre dos puntos.

Ejercicio 3

Diseñar una estructura para almacenar una fecha representada por tres números enteros. Hacer funciones para comprobar la validez de la fecha, convertir la fecha a la forma juliana (número de día, año), ejemplo: el 1 de febrero de 2010 se representa por 32 2010. Para representar la fecha en forma juliana diseñar otra estructura.

Ejercicio 4

Diseñar una estructura para almacenar los datos de un producto con código, descripción, precio unitario y stock.

Ejercicio 5

Diseñar una estructura para almacenar los datos de un círculo, otra para los datos de un rectángulo en la forma de base y altura. Hacer funciones para el cálculo de la superficie del círculo y del rectángulo y el perímetro del círculo del rectángulo.

Ejercicio 6

Modificar la estructura realizada en 5 para el rectangulo para definirlo como las coordenadas de un punto en el plano y el ancho y el largo, realizar las mismas funciones.

Ejercicio 7

Modificar la estructura de 5 y 6 para definir la rectángulo como dos puntos del plano. Volver a realizar las mismas funciones.

Trabajo práctico B-6

Trabajo integrador

Objetivos:

Que el alumnos sepa aplicar todos los conocimientos aprendidos durante la cursada de la asignatura.

Enunciado
Hacer una aplicación en C que administre una colección de datos. La aplicación deberá poder procesar: alta no existente, baja existente, consulta y modificación. También se deberá tener en cuenta que dentro de las consultas deberá administrar solicitudes de listado ordenados de tres diferentes modos: alfabéticamente, numéricamente y por fecha. La función de ordenar deberá ser única y se deberá pasar como parámetro la función de comparación.

Ejemplo de colecciones de datos:

1. productos de un supermercado con fecha de vencimiento
2. libros de una biblioteca con la fecha de la última edición
3. alumnos de una carrera con fecha de ingreso
4. autos de una concesionaria con la fecha de la fabricación
5. empleados de una empresa con fecha de ingreso
6. cuadros de un museo con la fecha en que fue realizada

El trabajo deberá usar arreglos de estructuras (por lo menos uno) y la estructura deberá constar del campo fecha.
El trabajo será realizado de manera individual y cada alumno deberá elegir un tema: no puede haber dos alumnos con el mismo tema.

Por ejemplo, en el caso 1) se podrá ordenar por código de producto, por cantidad en stock o por descripción (una de los dos) y por fecha de vencimiento.