Ejercicios de la asignatura Programación I

Grado en Ingeniería de Sistemas de Información
Primer curso
Profesor: Mariano Fernández López
Escuela Politécnica Superior, Universidad San Pablo CEU
15 de octubre de 2014

Índice

1. El catálogo de ofertas de ordenadores

Se pretende escribir un programa en Java que ayude a gestionar un catálogo de ofertas de ordenadores. Un prototipo de dicho programa se va a desarrollar siguiendo el guion de los ejercicios que se muestran a continuación.

Ejercicio 1. Defina la clase *ModeloOrdenador* con los atributos *marca*, *identificadorDeModelo*, *modelo de procesador*, *memoriaRam* y *discoDuro*.

Ejercicio 2. Añada al programa del ejercicio 1 una clase *OfertaOrdenador* con los atributos *precioInicial* y *descuento*, y el método *calcularPrecioFinal()*. Esta clase estará relacionada con la clase *ModeloOrdenador* creada anteriormente. Es decir, cada oferta se referirá a un modelo de ordenador.

Ejercicio 3. Escriba un método que escriba las características de cada modelo de ordenador.

Ejercicio 4. Escriba un método que escriba las características de cada oferta de ordenador.

Ejercicio 5. Escriba una clase *GestorOfertas* con un método estático de inicio llamado *inicio*.

Ejercicio 6. Escriba *getters* y *setters* para las clases implementadas anteriormente.

Ejercicio 7. Escriba un programa que permita saber si una oferta de ordenadores sigue en vigor.

Ejercicio 8. Escriba una clase *CatalogoDeOfertasDeOrdenadores* con una lista de ofertas de ordenadores. Dicha clase debe tener un método que permita escribir todas las ofertas.

Ejercicio 9. Programe el alta, baja y modificación de ofertas de ordenador utilizando la persistencia de objetos.

Ejercicio 10. Programe la generación de una hoja de cálculo con las ofertas del problema de las ofertas de ordenador.

Ejercicio 11. Comente el programa del catálogo de ofertas de ordenadores, y genere el *javadoc*.

2. Administración territorial

Ejercicio 12. Cree la clase *Localidad* con los atributos *nombre* y *numeroDeHabitantes*. A continuación, cree la clase *Municipio*. Un municipio está formado por localidades. Una de ellas (donde está el ayuntamiento) es la

cabeza del municipio. El número de habitantes del municipio es la suma de los habitantes de sus localidades. Luego, cree, la clase *Provincia*, que está formada por municipios, y uno de ellos es la capital. El número de habitantes de la provincia es la suma de los habitantes sus municipios.

Ejercicio 13. Extienda el ejercicio anterior para mostrar, añadir, modificar y borrar localidades, municipios y provincias.

Ejercicio 14. Extienda el ejercicio anterior para generar una hoja de cálculo con tres pestañas: una para las localidades, otra para los municipios y otra para las provincias.

3. Libreta de contactos

Ejercicio 15. Programe una libreta de contactos. Cada contacto consistirá en: nombre, teléfono, e-mail y dirección postal. Tendrá que ser posible mostrar, añadir, modificar y borrar contactos.

Ejercicio 16. Extienda el programa anterior para que sea posible mostrar, añadir, modificar y borrar los datos de la localidad de cada persona introducida en la libreta de contactos, por ejemplo, el nombre, número de habitantes, provincia, comarca, etc. de la localidad donde nació.

Ejercicio 17. Extienda el programa anterior para que genera una hoja de cálculo con dos pestañas: una con los datos en sí del contacto: nombre, e-mail, dirección postal y nombre de la localidad, y otra con los datos de cada localidad.

4. Plan de estudios

Ejercicio 18. Codifique la clase *Asignatura* con los atributos *nombre* y *ects* (número de créditos ECTS). A continuación, codifique la clase *Curso*, con los atributos *numeroDeCurso* y *asignaturas* (que representa la secuencia de asignaturas que forman un curso). Luego, implemente un método para mostrar un curso. La información que hay que escribir por pantalla es: el número de curso, las asignaturas (con su nombre y su número de créditos ECTS) así como la suma total de créditos ECTS del curso.

Ejercicio 19. Extienda el programa anterior con la clase *PlanDeEstudios*, que debe contener el atributo *nombre* (por ejemplo, *Grado en Ingeniería de Sistemas de Información*) y la secuencia de cursos. Después, implemente un método para mostrar el plan de estudios. La información a mostrar por pantalla es: el nombre del plan de estudios, cada uno de los cursos y la suma total de créditos ECTS del plan de estudios.

Ejercicio 20. Extienda el programa anterior para que genere una hoja de cálculo con el plan de estudios.

5. El fabricante de vehículos

Ejercicio 21. Un fabricante de vehículos ha encargado el desarrollo de un sistema para gestionar sus ventas. El primer prototipo permitirá mostrar en modo texto por pantalla, para cada petición de venta, la siguiente información:

- El código de la petición.
- El modelo del vehículo.
- El precio del vehículo con el equipamiento básico.
- Si se trata de un turismo, el número de plazas, y, si se trata de una furgoneta, el volumen de carga (en m^3).
- Para cada extra, su descripción y su precio.
- Finalmente, se mostrará el precio total, que será la suma del precio del vehículo más los extras.

A modo de ejemplo, se muestra un listado con dos peticiones de compra (con códigos PT-3456 y PT-3457 respectivamente):

```
PETICIÓN DE COMPRA: PT-3456
===========================
Modelo: Vehículo de turismo La Gallina Americana
Precio con equipamiento básico: 15000.0
Número de plazas: 5
Equipamiento extra:
     Sistema V2C, precio: 325.0
     F Bluetooth Music, precio: 599.0
Precio total del vehículo más los extras: 15924.0


PETICIÓN DE COMPRA: PT-3457
===========================
Modelo: Furgoneta La Pava
Precio con equipamiento básico: 25000.0
Volumen de carga: 5.2
Equipamiento extra:
     Sistema V2C, precio: 325.0
Precio total del vehículo más los extras: 25325.0
```

Los técnicos han decidido inicialmente crear las siguientes clases: `Vehiculo`, `Turismo`, `Furgoneta`, `Equipamiento`, `PeticionDeCompra` y la clase que contiene el método `main`.

Se pide implementar en Java:

1. La jerarquía de clases de vehículos con las clases: `Vehiculo`, `Turismo` y `Furgoneta`. Tales clases deberán incluir métodos que permitan mostrar, para cualquier vehículo, su modelo y su precio con el equipamiento básico. Además, si se trata de un turismo, deberá mostrar el número de plazas y, si se trata de una furgoneta, el volumen de carga.

2. La clase `Equipamiento`, que permitirá escribir, para cualquier extra, su descripción y su precio.

3. La clase `PeticionDeCompra`, que mostrará el código de la petición, la información del vehículo (haciendo uso de los métodos implementados en el apartado 1), los extras (haciendo uso de los métodos implementados en el apartado 2) así como el precio total del vehículo más los extras.

4. La clase que contiene el método `main` para generar el listado del ejemplo haciendo uso de los métodos implementados en los apartados anteriores.

6. La clase String

Ejercicio 22. Busque en Internet los métodos de la clase *String*.

Ejercicio 23. A partir de la siguiente cadena: *No vueles como las aves de corral cuando puedes subir como las águilas*. Se pide:

1. La cadena formada por los dos primeros caracteres.

2. El quinto carácter.

3. La posición de la palabra *puedes*.

4. Reemplazar *aves de corral* por *gallinas*.

7. Fórmulas matemáticas y físicas

En los siguientes ejercicios se debe elevar una excepción cuando los datos con los que se pretende realizar los cálculos no sean válidos (tiempos menores que cero, divisiones por cero, suma de vectores de diferentes tamaños, etc.).

Ejercicio 24. Programe una clase *SerieAritmetica* con atributos *a1* (valor el primer elemento de la serie) y d (diferencia entre cada término y el siguiente) que permita obtener la suma sus n primeros elementos.

Ejercicio 25. Programe una clase que permita, a partir de la velocidad inicial (v_0) de un móvil uniformemente acelerado, el tiempo (t) que hace que ha alcanzado tal velocidad, y su aceleración (a), calcular el espacio recorrido (e).

Recuérdese que la fórmula del espacio recorrido por un móvil uniformemente acelerado es $e = v_0 t + \frac{1}{2} a t^2$.

Ejercicio 26. Añada al programa del ejercicio 25 la posibilidad de, sabiendo su velocidad incial (v_0), su velocidad final (v_f) y el tiempo transcurrido entre ambas velocidades (t), obtener su acelaración.

Ejercicio 27. Programe una calculadora de vectores que permita realizar las siguientes operaciones:

1. Multiplicar un escalar por un vector.
2. Sumar dos vectores.
3. Calcular el producto escalar de dos vectores.

Ejercicio 28. Programe una calculadora de matrices que permita realizar las siguientes operaciones:

1. Multiplicar un escalar por una matriz.
2. Sumar dos matrices.

Ejercicio 29. Escriba un programa que calcule la integral definida $\int_0^3 2\, dx$.

Ejercicio 30. Escriba un programa que calcule la integral definida $\int_0^1 \frac{1}{\sqrt{2\pi}} e^{\frac{-1}{x^2}}\, dx$.

8. Empresa con empleados y consultores

Ejercicio 31. Implemente la clase *Empleado* con atributos *nombre*, *apellido1*, *apellido2*, *numeroDeTrabajador*, *nif*, *sueldo* y *retencionIrpf*, y con un método para calcular el sueldo final a partir del sueldo y de la retención de IRPF.

Ejercicio 32. Añada la clase *Consultor* con atributos *nombre*, *apellido1*, *apellido2*, *numeroDeTrabajador*, *nif*, *tarifaHoraria* y *horasTrabajadas*, y con un método para calcular la cantidad a pagar al consultor a partir de la tarifa horaria y el número de horas trabajadas. Es importante que no replique código.

Ejercicio 33. Añada la clase *Empresa* de tal forma que, para cada empresa, puedan trabajar tando consultores como empleados. Dicha clase tendrá un método que permita sumar la cantidad a abonar a todos los trabjadores, tanto empleados como consultores.

Ejercicio 34. Genere una hoja de cálculo con los datos de los trabajadores de la empresa del problema 33.

9. Análisis y reestructuración de programas

Ejercicio 35. Escriba la salida por pantalla del siguiente programa, y explique su respuesta:

```
package paquete1;

/**
 *
 * @author USUARIO
 */
public class Clase1 {
    int miMetodo(int a, int b)
    {
        return a + b;
    }
}

package paquete1;

/**
 *
 * @author USUARIO
 */
public class Clase2 extends Clase1{
    @Override
    int miMetodo(int a, int b)
    {
        return super.miMetodo(a, b) + 2;
    }

}

package paquete1;

/**
 *
 * @author USUARIO
 */
public class Clase3 extends Clase1{
    int miMetodo(int a, int b)
    {
        return super.miMetodo(a, b) + 3;
    }
}
package paquete1;

/**
 *
 * @author USUARIO
 */
public class Coordinador {

    /**
```

```
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Clase1[] objetos = new Clase1[3];

        objetos[0] = new Clase2();
        objetos[1] = new Clase3();
        objetos[2] = new Clase1();

        System.out.println(objetos[0].miMetodo(4, 5));
        System.out.println(objetos[1].miMetodo(4, 5));
        System.out.println(objetos[2].miMetodo(4, 5));
    }
}
```

Ejercicio 36. Identifique al menos cuatro tipos de errores en el siguiente programa.

```
package paquete3;

/**
 *
 * @author USUARIO
 */
public interface Interfaz1 {
    public int miMetodo11();
    public int miMetodo12(int a, int b);

}
package paquete3;

/**
 *
 * @author USUARIO
 */
public interface Interfaz2 {
   int miMetodo21(int a, int b);
   int miMetodo22(int a, int b);
}
public class Clase1 implements Interfaz1, Interfaz2{
    int miMetodo11()
    {
        System.out.println("Introduzca un número");

        int a = Integer.parseInt(br.readLine());

        System.out.println("Introduzca otro número");

        int b = Integer.parseInt(br.readLine());

        return a + b + 11;
    }
```

```
    int miMetodo21(int a, int b)
    {
        return a + b + 21;
    }

    int miMetodo22(int a, int b)
    {
        return a + b + 21;
    }
}
```

Ejercicio 37. ¿Qué escribe por pantalla el siguiente programa? Justifique la respuesta:

```
package paquete2;

/**
 *
 * @author USUARIO
 */
public abstract class Clase1 {
    int miMetodo1(int a, int b)
    {
        return a + b;
    }

    abstract int miMetodo2(int a, int b);
}
package paquete2;

/**
 *
 * @author USUARIO
 */
public class Coordinador {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        System.out.println((new Clase1()).miMetodo1(4, 5));
    }
}
```

Ejercicio 38. Sea el código que se muestra a continuación:

```
// Fichero 1
package profesores;
```

```
/**
 * Son los tipos de acreditación que puede tener un profesor de
    universidad.
 *
 * @author USUARIO
 *
 */
public enum Acreditacion {
    TITULAR_DE_UNIVERSIDAD,
    CATEDRÁTICO_DE_UNIVERSIDAD
}

// Fichero 2

package profesores;

/**
 * Son las categorías de profesor de universidad
 *
 * @author USUARIO
 */
public enum Categoria {
    COLABORADOR_NO_DOCTOR,
    COLABORADOR_DOCTOR,
    AGREGADO,
    CATEDRATICO
}

// Fichero 3

package profesores;

import java.util.Calendar;

/**
 * Se trata de la clase de profesores. Todo profesor tiene
 * un número de personal, dos apellidos, un nombre y una categoría
    .
 * La categoría del profesor puede ser: contratado no doctor,
 * contratado doctor, agregado o catedrático. Cualquiera de ellos,
     salvo el
 * contratado no doctor, tiene una fecha de lectura de tesis. El
    agregado
 * está acreditado como titular de universidad, y el catedrático
    como
 * catedrático de universidad. El catedrático tiene un número de
    tesis
 * doctorales dirigidas.
 *
 * @author USUARIO
 * @see Categoria
 * @see Calendar
 * @see Acreditacion
 */
```

```
public class Profesor {
    int numeroPersonal;
    String apellido1, apellido2, nombre;
    Categoria categoria;
    Calendar fechaLecturaTesis;
    private Calendar fechaAcreditacion;
    int nTesisDirigidas;

   /**
    * Transforma a String una fecha formada por día del mes, mes y
        año.
    *
    * @param fecha La fecha a transformar.
    * @return la cadena con los datos de la fecha.
    */
    public String presentacionFecha(Calendar fecha)
    {
        return fecha.get(Calendar.DAY_OF_MONTH)+"/"+
               fecha.get(Calendar.MONTH)+"/"+
               fecha.get(Calendar.YEAR);
    }


   /**
    * Construye un profesor a partir de los datos de entrada.
       Dependiendo de
    * la categoría, utilizará unos datos u otros en la construcci
       ón.
    *
    * @param datos Array que contiene, en este orden, numero de
       personal,
    * primer apellido, segundo apellido, nombre, categoría, fecha
       de lectura de
    * tesis, fecha de acreditación y número de tesis dirigidas.
    * apellido 2.
    * @return la cadena con los datos de la fecha.
    */
    public Profesor(Object... datos)
    {
        numeroPersonal = ((Integer) datos[0]).intValue();
        apellido1 = (String) datos[1];
        apellido2 = (String) datos[2];
        nombre = (String) datos[3];
        categoria = (Categoria) datos[4];
        switch(categoria)
        {
            case COLABORADOR_DOCTOR: case AGREGADO: case
                CATEDRATICO:
                fechaLecturaTesis = (Calendar) datos[5];
                if (categoria == Categoria.AGREGADO
                    categoria == Categoria.CATEDRATICO)
                        fechaAcreditacion = (Calendar) datos[6];
                if (categoria == Categoria.CATEDRATICO)
                        nTesisDirigidas = ((Integer) datos[7]).
```

```
                        intValue();
    }
}

/**
 * Devuelve en un String, según la categoría del profesor, sus
    datos
 * correspondientes.
 *
 * @return la cadena con los datos del profesor.
 */
public String presentacionProfesor()
{
    String cadenaComun = "Número identificador de personal: "
        +
            numeroPersonal + "\n" +
            "Apellidos: " + apellido1 + " " + apellido2 + "\n"
                +
            "Nombre: " + nombre + "\n";
    String cadenaLecturaTesis = null;
    String cadenaFechaAcreditacion = null;

    String cadenaNTesisDirigidas = "Número de tesis dirigidas:
         " +
                                    nTesisDirigidas + "\n";

    if (categoria == Categoria.COLABORADOR_DOCTOR
        categoria == Categoria.AGREGADO
            categoria == Categoria.CATEDRATICO)
                cadenaLecturaTesis = "Fecha lectura de tesis:
                    " +
                        presentacionFecha(fechaLecturaTesis);

     if (categoria == Categoria.AGREGADO
            categoria == Categoria.CATEDRATICO)
                cadenaFechaAcreditacion = "Fecha de acreditaci
                    ón: "
                                    + presentacionFecha(
                                        fechaAcreditacion);


    switch(categoria)
    {
        case COLABORADOR_NO_DOCTOR: return cadenaComun +
                                      "Categoría: colaborador
                                          no doctor\n";
        case COLABORADOR_DOCTOR: return cadenaComun +
                                    "Categoría: colaborador
                                        doctor\n" +
                                    cadenaLecturaTesis + "\n"
                                        ;

        case AGREGADO: return cadenaComun + "Categoría:
            agregado\n" +
```

```
                            cadenaLecturaTesis + "\n" +
                             "Tipo de acreditación: Titular de
                                 Universidad\n" +
                            cadenaFechaAcreditacion + "\n";
            case CATEDRATICO: return cadenaComun + "Categoría:
                catedrático\n" +
                            cadenaLecturaTesis + "\n" +
                        "Tipo de acreditación: Catedrático de
                             Universidad\n" +
                            cadenaFechaAcreditacion + "\n" +
                            cadenaNTesisDirigidas + "\n";
            default: return null;
        }
    }

    Acreditacion getAcreditacion()
    {
        switch(categoria)
        {
            case AGREGADO: return Acreditacion.
                TITULAR_DE_UNIVERSIDAD;
            case CATEDRATICO: return Acreditacion.CATEDR
                ÁTICO_DE_UNIVERSIDAD;
            default: return null;
        }
    }
}

// Fichero 4
package profesores;

import java.util.Calendar;

/**
 *
 * Se trata de una clase para comprobar que las otras clases
     funcionan más
 * o menos.
 *
 * @author USUARIO
 */
public class Coordinador {

    /**
     * Es la clase de entrada al programa
     *
     * @param args no se utilizan realmente.
     *
     */
    public static void main(String[] args) {
        // TODO code application logic here

        Universidad universidad = new Universidad("Universidad del
             Sur", 4);
```

```
        Calendar fechaLecturaTesis = Calendar.getInstance();
        Calendar fechaAcreditacion = Calendar.getInstance();

        universidad.tieneProfesores[0] =
                new Profesor(1, "García", "García", "Antonio",
                                    Categoria.
                                        COLABORADOR_NO_DOCTOR);
        //System.out.println(profesor1.presentacionProfesor());

        fechaLecturaTesis.set(2000, 10, 21);
        universidad.tieneProfesores[1] =
                new Profesor(2, "Pérez", "García", "Juan",
                               Categoria.COLABORADOR_DOCTOR,
                                   fechaLecturaTesis);
        //System.out.println(profesor2.presentacionProfesor());

        fechaLecturaTesis.set(1998, 11, 22);
        fechaAcreditacion.set(2001, 8, 3);
        universidad.tieneProfesores[2] =
                new Profesor(3, "Rodríguez", "Martínez", "Pedro",
                                    Categoria.AGREGADO,
                                        fechaLecturaTesis,
                                    fechaAcreditacion);
        //System.out.println(profesor3.presentacionProfesor());

        fechaLecturaTesis.set(1980, 2, 13);
        fechaAcreditacion.set(1997, 7, 5);
        universidad.tieneProfesores[3] =
                new Profesor(4, "Mata", "Rubio", "José",
                                    Categoria.CATEDRATICO,
                                        fechaLecturaTesis,
                                    fechaAcreditacion, 7);
        //System.out.println(profesor4.presentacionProfesor());

        System.out.println(universidad.presentacionUniversidad());
    }
}
// Fichero 5
package profesores;

/**
 * Sobre la universidad nos interesa saber su nombre y sus
     profesores.
 * @author USUARIO
 */
public class Universidad {
    public int nProfesores;
    String nombre;
    public Profesor[] tieneProfesores;

    /**
    * Construye una universidad a partir de su nombre y su número
        de profesores.
```

```
 * @param nombre Es el nombre de la universidad.
 * @param nProfesores Es el número de profesores de la
    universidad.
 * @see Profesor
 */
 public Universidad(String nombre, int nProfesores)
 {
     this.nombre = nombre;
     this.nProfesores = nProfesores;
     tieneProfesores = new Profesor[nProfesores];
 }

/**
 * Devuelve en un String con el nombre de la universidad y los
    de sus
 *profesores
 *
 * @return la cadena con la información de la universidad.
 * @see Profesor
 */
 public String presentacionUniversidad()
 {
     String cadena = nombre + "\n";
     for(int i=0; i < nProfesores; i++)
     {
         cadena = cadena + "\n" + tieneProfesores[i].
             presentacionProfesor()
                 +"\n";
     }

     return cadena;
 }
 }
```

Se pide escribir un programa que realice la misma función; pero con un software de más calidad.

10. Un prototipo de aplicación informática en el dominio de las ciencias de la vida

Ejercicio 39. Sean los siguientes escenarios:

Escenario 1. Generación de hoja de cálculo a partir de compuestos químicos. El usuario solicita la generación de una hoja de cálculo con los compuestos químicos almacenados en el sistema, y el sistema la genera en la ruta:

`recursos/compuestos.xls`

con un contenido similar al del cuadro 1.

Escenario 2. Generación de hoja de cálculo a partir de vías metabólicas. El usuario solicita la generación de una hoja de cálculo de las vías metabólicas almacenadas en el sistema, y el sistema la genera en la ruta:

Cuadro 1: Vista parcial de la hoja de cálculo generada en el escenario 1

Id	**Nombre**	**Masa**	**Fórmula**
C16408	2',3,4,4',6'-Peptahydroxychalcone 4'-O-glucoside	450,1162	C21H22O11
C16408	PHC 4'-O-glucoside	450,1162	C21H22O11
C16408	2',3,4,4',6'-Pentahydroxychalcone 4'-O-beta-D-glucoside	450,1162	C21H22O11
C00353	Geranylgeranyl diphosphate	450,1936	C20H36O7P2
C00353	Geranylgeranyl pyrophosphate	450,1936	C20H36O7P2
C00353	all-trans-Geranylgeranyl diphosphate	450,1936	C20H36O7P2
C00353	all-trans-Geranylgeranyl pyrophosphate	450,1936	C20H36O7P2
C06089	ent-Copalyl diphosphate	450,1936	C20H36O7P2
C06089	(-)-Copalyl diphosphate	450,1936	C20H36O7P2
C11356	trans,trans,cis-Geranylgeranyl diphosphate	450,1936	C20H36O7P2

`recursos/vias.xls`

con un contenido similar al del cuadro 2.

Cuadro 2: Vista parcial de la hoja de cálculo generada en el escenario 2

Vía metabólica	**Compuesto**
Glycolysis / Gluconeogenesis	2',3,4,4',6'-Peptahydroxychalcone 4'-O-glucoside
Glycolysis / Gluconeogenesis	PHC 4'-O-glucoside
Glycolysis / Gluconeogenesis	2',3,4,4',6'-Pentahydroxychalcone 4'-O-beta-D-glucoside
Glycolysis / Gluconeogenesis	Geranylgeranyl diphosphate
Citrate cycle (TCA cycle)	3alpha,7alpha,12alpha-Trihydroxy-5beta-cholestanoate
Citrate cycle (TCA cycle)	3alpha,7alpha,12alpha-Trihydroxy-5beta-cholestan-26-oate
Citrate cycle (TCA cycle)	3alpha,7alpha,12alpha-Trihydroxy-5beta-cholestanate
Citrate cycle (TCA cycle)	Phylloquinone
Citrate cycle (TCA cycle)	Vitamin K1
Pentose phosphate pathway	2',3,4,4',6'-Peptahydroxychalcone 4'-O-glucoside
Pentose phosphate pathway	PHC 4'-O-glucoside
Pentose phosphate pathway	2',3,4,4',6'-Pentahydroxychalcone 4'-O-beta-D-glucoside
Pentose phosphate pathway	Geranylgeranyl diphosphate
Pentose phosphate pathway	Geranylgeranyl pyrophosphate
Pentose phosphate pathway	all-trans-Geranylgeranyl diphosphate
Pentose phosphate pathway	all-trans-Geranylgeranyl pyrophosphate

Escenario 4. CRUD. El usuario puede insertar, modificar o borrar tanto compuestos químicos como vías metabólicas. Se pide:

1. Diagrama de clases UML válido para los dos escenarios anteriores. Puede omitir las clases de la interfaz de usuario.

2. Implementación en Java del siguiente método así como del resto de métodos y de clases en que necesite apoyarse:

```
/**
* Se genera una hoja de cálculo con los datos de la lista
     de elementos y se guarda
```

```
* en la ruta recursos/nombreFichero
*
* @param elementos es la lista de datos que debe aparecer
     en la hoja de cálculo
* @param nombreFichero es el nombre del fichero de la
    hoja de cálculo
*/
public void generarHoja(List elementos, String
    nombreFichero)
```

Este método permite generar tanto la hoja de cálculo para los compuestos como la hoja de cálculo para las vías metabólicas. Los elementos (en el sentido de los elementos de una lista, no de elementos químicos) son objetos persistentes que se han recuperado para generar la hoja de cálculo.

IMPORTANTE. En la valoración del ejercicio se penalizará, entre otras deficiencias, el abuso de estructuras de control y la semejanza en el código de métodos diferentes ("copia y pega").