Problema 1

Escriba en Java una clase genérica `Conjunto<G>` que permita representar conjuntos de elementos de cualquier tipo (el mismo elemento no puede estar más de una vez en el conjunto). Las características básicas que se deben implementar son las siguientes:

- `public void agrega(G elemento)`
- `public void elimina(G elemento)`
- `public boolean esta(G elemento)`
- `public boolean esVacio()`
- `public int getCardinal()`

Del mismo modo, se deben implementar las funciones que permitan obtener la unión, la intersección y la diferencia de dos conjuntos para obtener un tercero. Diseñe cómo sería la signatura de esos métodos

Notas: Puede utilizar la clase `ArrayList<G>` como clase auxiliar.

Problema 2

Crear una clase que permita modelar números complejos. Elaborar una clase `Complejo` que modele los números complejos implementando al menos las operaciones de suma, resta y módulo de un número complejo así como la multiplicación del número complejo por un número real.

Problema 3

Escriba una clase genérica `Vector` que represente vectores de números. Debe implementar: la suma y resta de vectores del mismo tamaño, la multiplicación y división por un número real para dar como resultado un nuevo vector.
Combine el Problema 2 y el Problema 3 de forma que la clase genérica `Vector` que se ha programado también se pueda utilizar con los números complejos que describe la clase `Complejo` realizada en el Problema 2. Modifique la clase `Complejo` si no puede ser utilizada tal cual estaba.

Problema 4

Elaborar una clase Polinomio que modele los polinomios de grado dos implementando al menos las operaciones de suma obtener un tercer polinomio y producto por un número y el cálculo de las raíces reales del polinomio, si es que existen.

Problema 5

Elaborar una clase Racional que modele los números racionales, implementando al menos suma, opuesto, producto e inverso del racional. El racional se debe almacenar en forma reducida.
Nota 1: Decimos que una fracción no nula está en forma reducida cuando el numerador y el denominador no tienen divisores comunes. La forma reducida de la fracción nula es 0/1

Nota 2: Utilice métodos de la clase Math para facilitar la tarea.

Problema 6

Escribir una clase `Reloj` que simule el comportamiento de un reloj digital (con las características reset que pone el reloj a 00:00:00, poner en hora, incrementar que avanza el reloj un segundo, etc.). Se quiere redefinir el método `toString()` heredado de `Object` para poder obtener una representación del reloj digital en cadena de caracteres. La clase debe garantizar que el tiempo marcado es siempre correcto (entre 00.00.00 y 23.59.59).

Problema 7

Elaborar una clase que modele una fecha. La clase deberá disponer de características que devuelvan el día, el mes y el año, además de una implementación del método `toString()` que obtiene una cadena con la fecha en forma abreviada (16/02/2015) y extendida (16 de febrero de 2015). Para obtener una u otra cadena la clase deberá tener un método `setExtendida()` y un método `setAbreviada()`. La clase deberá disponer de una función llamada incremento, con un parámetro entero, que fabrique una nueva fecha, resultado de incrementar la original en el número de días dado por el parámetro. En este ejercicio no utilice las clases de la plataforma para el manejo de fechas.

Nota$_1$: Son años bisiestos los múltiplos de 4 que no lo son de 100, salvo que sean múltiplos de 400 en cuyo caso sí son bisiestos.

Nota$_2$ Para la solución de este problema puede ser útil definir un método incrementoUnDia·

Problema 8

Implemente un sistema formado por dos clases que simularán una antigua sumadora de enteros. La sumadora dispone de un acumulador y un valor de entrada de ocho dígitos cada uno, e implementa las siguientes funciones:

Poner a cero el acumulador y la entrada [Tecla c].

Añadir un dígito (0..9) [teclas numéricas] al número de la entrada por la derecha [actualizando el número visualizado]. Cuando la entrada tiene ya todas las cifras ocupadas esta orden será ignorada por la sumadora.

Sumar [tecla +] el número del acumulador con el de la entrada poniendo a cero el número de entrada [y el número visualizado].

Sumar [tecla =] el número del acumulador con el de la entrada, poniendo éste último a cero [y mostrando el resultado mediante el número visualizado].

[Cualquier entrada diferente de estas será ignorada].

Si, al realizar la suma, se produce un error de desbordamiento la máquina queda bloqueada [mostrando en el display el mensaje error hasta recibir la orden c]

Se debe separar completamente el diseño de la sumadora y el de la interfaz de usuario [lo que aparece entre corchetes], por lo que debe utilizar dos clases como mínimo (Sumadora e InterfazConsolaSumadora) así como una clase Lanzadora con un método main para crear una sumadora, una interfaz por consola y asociarlas y comenzar a escuchar las pulsaciones de teclas.

Problema 9

Elaborar una clase que sirva de modelo para los vectores de R^3. La clase debe proporcionar como servicios la constante «origen» y las siguientes operaciones (que proporcionan un nuevo vector de R^3):

suma: (x,y,z)+(x',y',z')=(x+x',y+y',z+z')
producto: (x,y,z) * (x',y',z') = (y*z'-z*y',z*x'-x*z',x*y'-y*x') producto por un real: a * (x,y,z) = (a*x,a*y,a*z)

Si has hecho la clase genérica Vector del Problema 3, no merece la pena que hagas esta.

Problema 10

Dentro de un sistema bancario que ya está funcionando disponemos de una clase Cuenta definida del modo que sigue:

```
public class Cuenta {
  // parte privada de la clase

   public Cuenta(Cliente titular)  { … }

   public Cliente getTitular() { ... } // Cliente tiene un método getEdad

   public float getSaldo() { … }

   // otros métodos

   /**
   ** @assertion.pre (cantidad >=0 && getSaldo()-cantidad>=0)
   **/
   public void reintegro(float cantidad) { … }


   /**
   ** @assertion.pre (cantidad >=0)
   **/

   public void ingreso(float cantidad) {}
}
```

La nueva política del banco exige que tengamos otros tipos de cuentas con características diferentes. En particular debemos admitir los siguientes tipos:

- **Cuenta de crédito:** Permite obtener reintegros hasta un saldo negativo fijo para cada cuenta y el banco cobra una comisión fija cada mes.
- **Cuenta de nómina:** Permite obtener reintegros hasta un saldo negativo igual al importe mensual de la nómina que el cliente tiene domiciliada en dicha cuenta. Se paga un interés todos los meses (pactado para cada cuenta) siempre que el saldo sea positivo. El banco cobra una comisión fija cada mes.
- **Cuenta de ahorro:** No admite saldo negativo. Cada mes con saldo al final superior a 1000€ se

ingresa el interés pactado (diferente para cada cuenta).

- **Cuenta joven:** Es una cuenta de ahorro sin saldo mínimo para el abono de intereses, pero exige que el cliente sea menor de 25 años.

  Implemente las clases necesarias y sus relaciones para solucionar las nuevas necesidades de cuentas detectadas.

Problema 11

En una aplicación de venta y configuración de ordenadores personales se considera un ordenador como la suma de varios componentes (una unidad central y varios elementos periféricos). El mínimo imprescindible para que se considere un ordenador es la unidad central, un dispositivo de entrada y otro de salida, pero pueden añadirse todos los que deseemos ofertar o que nos pidan los clientes. Para crearlo habrá que dar esos componentes mínimos y se puede modificar la configuración en cualquier momento añadiendo, quitando o cambiando exclusivamente periféricos. Por último, el precio de venta del ordenador es la suma de sus componentes.

Todos los componentes tienen información sobre el nombre del fabricante, el modelo y el precio de venta, que cambia con frecuencia.

Los dispositivos de entrada que manejamos actualmente son el teclado y el ratón. En todos los casos necesitamos saber el tipo de conector que utiliza (será un String) y los puertos válidos (varios valores de tipo entero).

Los dispositivos de salida de que disponemos son las pantallas y las impresoras (de inyección y láser). También tenemos que saber los puertos válidos. Además para las impresoras necesitamos saber el tipo de cartucho o "tóner" utilizado y el número de páginas impresas desde el último cambio de "tóner" (sólo para impresoras láser).

Implemente las clases que solucionan este problema

Problema 12

Tenemos una clase Empleado

```
public class Empleado {

    public Empleado(String nombre, int edad, String nif) {…}

    … y los getters de nombre, edad y nif

    @Override

        public String toString() { ...}

}
```

Al añadir nuevas capacidades al sistema descubrimos que necesitamos modelizar nuevos tipos de empleados:

- **Empleado temporal**, del que nos interesa saber la fecha de alta y de baja en la empresa.

- **Empleado por horas**. Nos interesa el precio de la hora trabajada, y el número de horas que ha trabajado este mes. El primero es un dato fijo, mientras el segundo varía todos los meses.
- **Empleado fijo.** Debemos añadir a la información que almacenamos sobre él el año de alta en la empresa.

  Además debemos añadir a todos los empleados la funcionalidad de cálculo del sueldo con las siguientes consideraciones:

- En los empleados temporales el sueldo mensual es fijo.
- En los empleados fijos el sueldo es el resultado de sumarle a la base un complemento anual fijo multiplicado por el número de años en la empresa.
- En los empleados por horas el sueldo se calcula multiplicando su sueldo por hora por el número de horas de este mes.

  Implemente las clases necesarias para solucionar las nuevas necesidades detectadas.

Problema 13

Dentro de un sistema de crédito en funcionamiento tenemos una clase Cliente de la que disponemos de su interfaz pública

```
public class Cliente {
        public Cliente(Persona persona) { … }
        public Persona getPersonaCliente() {…}
        public float getSaldoDispuesto() { …}
        public float getCargoAutorizado() {…}
        public void cargar(float importe) {…}
}
```

La nueva política de la empresa obliga a distinguir varios tipos de cliente además de incorporar un método que devuelva los tres pagos en que se fracciona el "saldo dispuesto", aplicando los siguientes criterios:

- **Cliente con descubierto.** No requiere ningún tipo de autorización para cargar un pago en la cuenta. Los tres pagos se calculan como el 50% del saldo dispuesto el primero, y el 25% del saldo dispuesto los otros dos.
- **Cliente con descuento fijo.** Al cargar un pago se le aplica un tanto por ciento fijo de descuento. Los tres pagos son una tercera parte del saldo dispuesto.
- **Cliente con descuento variable.** Se le aplica un descuento variable que debe indicarse para el cliente. Los tres pagos se calculan del mismo modo que en el cliente con descuento fijo.

Se pide:

- Definir las clases necesarias para satisfacer las nuevas necesidades detectadas, sin modificar la clase Cliente original

Problema 14

En una biblioteca universitaria multimedia se está construyendo un sistema de control de los fondos disponibles que incluyen documentos de distinto tipo. La forma de acceso es doble: Consultas en sala y préstamos temporales.

La consulta en sala requiere registrar el número del documento, la fecha y el DNI del alumno. El procedimiento de préstamo, en su caso, incluye los siguientes pasos: reservar el documento, recogerlo y devolverlo. Se manejan los mismos datos que en el caso de la consulta.

Los tipos de documentos que se contemplan son:

- Los libros clásicos, en papel. Los datos que interesa conocer son: Título, autor o autores, editorial, año de publicación. Estos libros se pueden prestar a los alumnos, salvo excepciones (diccionarios, normas ISO, etc., que sólo se pueden consultar en la sala).
- Las revistas en papel, que tienen las mismas características que los libros, más algunas peculiaridades: volumen, número y mes de salida. Se pueden consultar y prestar a los alumnos.
- Documentos en formato CD (libros, software). Se pueden prestar, al igual que los libros. En este caso interesa mantener algún dato más (formato del CD, tipo de licencia.).
- Revistas de investigación microfilmadas, que tienen las mismas características que las revistas en papel pero no se prestan y sólo se pueden consultar en la sala mediante terminales. Como dato adicional hay que mantener el código de microfilm.

  Diseñe las clases y relaciones que representen una solución para este problema. Se pide en concreto, la estructura de herencia implicada con el detalle de características atribuidas a cada clase y sus posibles redefiniciones. Escriba en Java las clases con los tipos de documentos y la interfaz de la clase que incluya los procedimientos de préstamo.

Problema 15

En un puerto se alquilan amarres para barcos de distinto tipo. Para cada Alquiler se guarda el nombre y DNI del cliente, las fechas inicial y final de alquiler, la posición del amarre y el barco que lo ocupará. Un Barco se caracteriza por su matrícula, su eslora en metros y año de fabricación.

Un alquiler se calcula multiplicando el número de días de ocupación (incluyendo los días inicial y final) por un módulo función de cada barco (obtenido simplemente multiplicando por 10 los metros de eslora) y por un valor fijo (12€ en la actualidad).

Sin embargo ahora se pretende diferenciar la información de algunos tipos de barcos:

- número de mástiles para veleros
- potencia en CV para embarcaciones deportivas a motor
- potencia en CV y número de camarotes para yates de lujo.

  El módulo de los barcos de un tipo especial se obtiene como el módulo normal mas:

  - el número de mástiles para veleros
- la potencia en CV para embarcaciones deportivas a motor
- la potencia en CV más el número de camarotes para yates de lujo.

Utilizando la herencia de forma apropiada, diseñe el diagrama de clases y sus relaciones, con detalle de atributos y métodos necesarios. Programe en Java las clases y los métodos que permitan calcular el alquiler de cualquier tipo de barco.

Problema 16

En un sistema de alarma de un edificio se consideran detectores de humo, sensores de temperatura, sensores de presión, etc. Todos estos elementos tienen un estado conectado/desconectado y en consecuencia se puede pasar de un estado a otro (cuando se crean, están desconectados). Todos ellos son capaces de proporcionar una medida (un valor real) y tienen un valor umbral que se fija inicialmente al crear el elemento. El sistema recorre en un bucle continuo todos sus elementos conectados. Cuando la medida de uno de ellos supera su valor umbral el sistema dispara la alarma.

Para evitar falsas alarmas, varios elementos se pueden unir formando grupos de sensores (y estos grupos a su vez se pueden unir formando otros grupos) y para este sensor complejo, la alarma sólo se dispara si el valor medio de los elementos del grupo supera el umbral definido para ese elemento compuesto. Implemente al menos todo lo relacionado con el disparo de la alarma.

Problema 17

Una empresa se organiza en una jerarquía de unidades de negocio. Para este caso, de la empresa nos interesa el presidente, el CIF y la dirección postal, mientras que de cada unidad de negocio nos interesa el gerente, el número de empleados, los beneficios brutos del último trimestre, la inversión en edificios y el número medio de contratos realizados por semana.

Además una unidad de negocio puede estar formada por varias unidades de negocio. En este caso empleados, beneficios e inversiones se obtienen como la suma de los datos correspondientes a las unidades de negocio que la componen, mientras que el número medio de contratos es la media de los números medios de éstas unidades de negocio.

Implemente las clases necesarias y dibuje el diagrama de clases con la solución.