
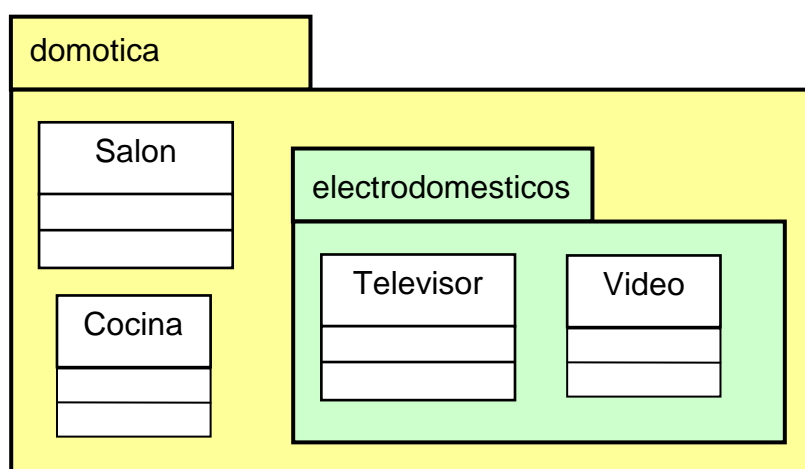


 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			 AENOR <b>E</b> Empresa Registrada ER-0934 / 2008 CERTIFIED <b>Net</b> QUALITY SYSTEM
<b>FP Parcial Presencial</b>	Pág. 1 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

## DESARROLLO DE CLASES Y UTILIZACIÓN DE OBJETOS

### 1. © Creación de Paquetes

- a. Abre un nuevo proyecto llamado 'Paquetes' en el que vamos a crear el siguiente diagrama:



Para ello seguiremos los siguientes pasos:

- Crearemos un paquete en el proyecto actual: Menú Archivo → Nuevo → Paquete → Nombre: *domotica*
- Ahora repetimos la operación para el paquete de nombre *domotica.electrodomesticos*
- A continuación crearemos las **clases vacías**: Salon, Cocina, Televisor y Video cada una dentro de su paquete.

Para visualizar la estructura de carpetas y ficheros que se ha creado, abrimos la carpeta **Mis Documentos\Java\Paquetes\**

 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			
<b>FP Parcial Presencial</b>	Pág. 2 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

## 2. © Proyecto Televisor

- a. Abre un nuevo proyecto llamado ‘*Televisor*’. A continuación crea y teclea la clase Televisor descrita a continuación:

Televisor
+ marca: String + modelo: String + anio: int // entero entre 1950 y 2200 + panoramica: boolean + stereo: boolean + encendida: boolean + volumen: int // entero entre 0 y 100 + canal: int // entero entre 0 y 99
+ void encender() + void apagar () + void seleccionarCanal (int nuevoCanal) + int obtenerCanal () + void subirCanal () + void bajarCanal () + void cambiarVolumen (int nuevoVolumen) + void imprimirCaracterísticas ()

Sabiendo que:

- El método “imprimirCaracterísticas” muestra en pantalla el valor de todas las propiedades del objeto.
- Si al cambiar el volumen o el canal de la televisión, el nuevo valor está fuera del rango permitido se debe mostrar un mensaje de error y dejar el valor que hubiera previamente.
- Si se intenta encender o apagar dos veces consecutivas el televisor no se debe mostrar ningún mensaje.
- Si el televisor está apagado sólo podremos encenderlo o imprimir sus características. En el resto de los métodos escribiremos: “Televisor apagado”
- Para el resto de los casos de los métodos se debe mostrar un mensaje que indique la acción que se ha realizado.



- b. Añadir a la clase anterior un método constructor con el siguiente prototipo:

- public Televisor (String marcaInicial, String modeloInicial, int aniolNi)

 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			 <small>ER-0934 / 2008</small>
<b>FP Parcial Presencial</b>	Pág. 3 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

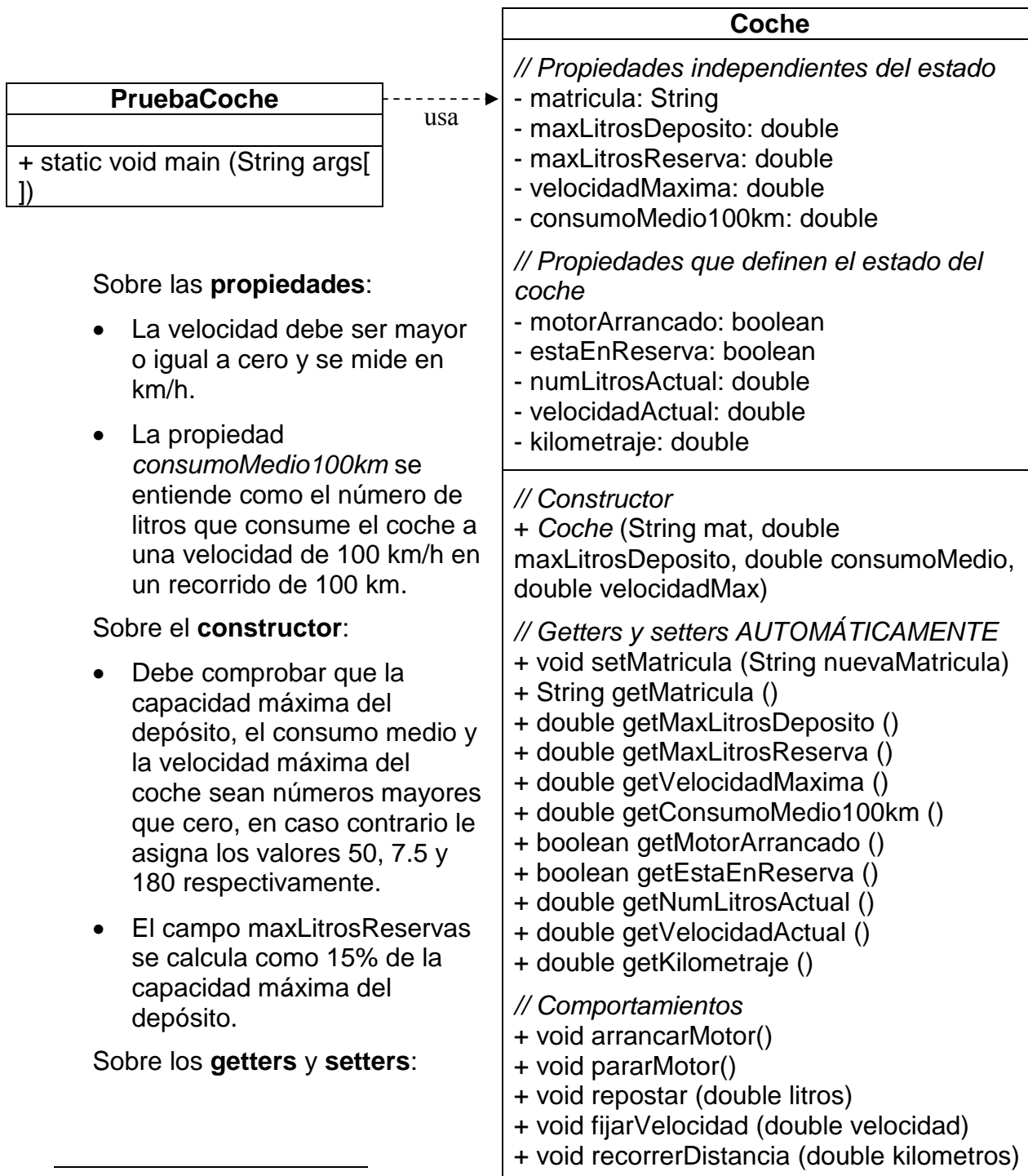
Sabiendo que:

- Si el valor de inicio del año se sale del rango permitidos entonces se le pone 2000.
  - El resto de las propiedades deben inicializarse con los valores por defecto que pone el compilador.
- c. Añadir una clase llamada *PruebaTelevisor* al proyecto del ejercicio anterior que debe verificar el buen funcionamiento de los objetos de tipo Televisor. Para ello haremos la siguiente secuencia de acciones:
- Crearemos un televisor de la marca “Sony”, modelo “Trinitron 4” del año 2003.
  - A continuación imprimiremos sus características.
  - Encendemos el televisor e imprimimos el canal seleccionado.
  - Ahora bajamos de canal (debe dar un mensaje de error)
  - Seleccionamos el canal 23 y después subimos de canal.
  - Ahora intentamos poner el volumen a 300 (debe dar un error).
  - Ponemos el volumen a 50 e imprimimos las características de nuevo.
  - Apagamos el televisor
  - Seleccionamos el canal 60 (debe decir que está apagado)
  - Y por último apagamos el televisor otra vez.

 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			
<b>FP Parcial Presencial</b>	Pág. 4 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

### 3. © Proyecto Coche

- a. Abre un nuevo proyecto llamado 'Coche' en el que vamos a crear las clases del siguiente diagrama:



Sobre las **propiedades**:

- La velocidad debe ser mayor o igual a cero y se mide en km/h.
- La propiedad *consumoMedio100km* se entiende como el número de litros que consume el coche a una velocidad de 100 km/h en un recorrido de 100 km.

Sobre el **constructor**:

- Debe comprobar que la capacidad máxima del depósito, el consumo medio y la velocidad máxima del coche sean números mayores que cero, en caso contrario le asigna los valores 50, 7.5 y 180 respectivamente.
- El campo *maxLitrosReservas* se calcula como 15% de la capacidad máxima del depósito.

Sobre los **getters** y **setters**:

 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			
<b>FP Parcial Presencial</b>	Pág. 5 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

- Se deben **generar automáticamente**. Para ello primero hay que declarar las propiedades y después seleccionamos:

*Menú Código fuente → Generar métodos de obtención y establecimiento*  
y marcamos las opciones que queramos.

Sobre los **comportamientos**:

- Si un método debe escribir algún mensaje en la pantalla, éste comenzará con el encabezado: **“El coche con matrícula *matricula bla bla blá*”** de esta forma podremos manejar varios objetos coches y los podremos distinguir por su matrícula.
- **Arrancar** el motor:
  - Sólo se arranca si queda algo de combustible, en cuyo caso escribe en pantalla: El coche con matrícula *matricula* ha arrancado.
  - Además si está en reserva debe escribir: El coche con matrícula *matricula* está en reserva de combustible
  - Si no queda combustible o ya estaba arrancado se debe escribir un mensaje que describa la situación.
- **Parar** el motor:
  - Si ya estaba en marcha se para y se escribe un mensaje diciendo que se ha parado el motor.
  - Si no estaba en marcha no hacemos nada.
- **Repostar** combustible:
  - Si lo que queda en el depósito más los litros que se echan superan la capacidad del tanque entonces llenamos el tanque hasta arriba y se muestra el mensaje: El coche con matrícula *matricula* ha rebosado el depósito.
  - En caso contrario se añaden los litros a los que ya había en el tanque.
  - Si el parámetro de los litros es negativo no cambiamos nada.
  - En cualquier caso el método debe imprimir el mensaje: El coche con matrícula *matricula* tiene *numero* litros de combustible.
- **Fijar velocidad**:

 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			
<b>FP Parcial Presencial</b>	Pág. 6 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

- Solo cambiamos podemos fijar la velocidad del coche si el motor está arrancado, en caso contrario mostramos un mensaje que describa la situación.
- Si la velocidad es mayor que la velocidad máxima del coche, entonces la fijamos a la velocidad máxima.
- Si la velocidad es negativa la ponemos a cero.
- Si cambiamos la velocidad debemos escribir un mensaje en pantalla.
- **Recorrer distancia:**
  - Si el motor está parado o bien está arrancado pero la velocidad es 0 entonces mostramos un mensaje y terminamos.
  - Si la distancia es negativa o cero mostramos un mensaje de error y terminamos.
  - En caso contrario tenemos que preguntarnos si con el combustible que quede y a la velocidad que hemos fijado somos capaces de recorrer tal distancia o nos quedamos sin combustible.
  - Para ello vamos a suponer que si voy a 110 km/h consumo un 10 % más del valor de la propiedad *consumoMedio100km* de esta forma podríamos calcular el **consumo instantaneo** del coche así:
$$\text{consumoMedio100km} * (1 + (\text{velocidadActual} - 100) / 100)$$
  - Ahora nos preguntamos cuántos litros harían falta para cubrir esa distancia yendo a esa velocidad. Para ello declaramos otra variable *litrosNecesarios* cuyo valor se calcula así:
$$\text{litrosNecesarios} = \text{distancia} * \text{consumoInstantaneo} / 100$$
  - Si el número de litros necesarios es menor que lo que queda en el depósito entonces podemos recorrer la distancia sin problemas. Actualizamos el kilometraje y los litros del depósito y escribimos el mensaje:

El coche con matricula *matricula* ha recorrido *distancia* kilómetros

Además si el coche ha entrado en reserva después del consumo de combustible hay que cambiar la propiedad *estaEnReserva*
  - En caso contrario nos vamos a quedar sin combustible en un determinado punto del trayecto. La **distancia real recorrida** se calcula como:
$$\text{distanciaReal} = 100 * \text{numLitrosActual} / \text{consumoInstantaneo}$$
  - Solo nos queda añadir esa distancia al kilometraje del coche, poner el depósito a cero, marcar que hemos entrado en reserva y mostrar los siguientes mensajes:

 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			
<b>FP Parcial Presencial</b>	Pág. 7 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

El coche con matrícula *matricula* ha recorrido *distancia* kilómetros

El coche con matrícula *matricula* esta sin combustible

El coche con matrícula *matricula* esta parado.

**b. La clase PruebaCoche:**

- Escribir un método main que cree un coche con matrícula “5466-FNZ”, con 60 litros de capacidad de depósito, con un consumo medio de 7.1 litros y con una velocidad máxima de 200 km/h.
- Después escribir el código necesario para repostar 15 litros de combustible, arrancarlo, fijar la velocidad a 80 km/h y recorrer 10 km.
- Por último, fijar la velocidad a 120 km/h e intentar recorrer 300 km.

 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			
<b>FP Parcial Presencial</b>	Pág. 8 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

## 4. Proyecto Geométrico: Clases Punto, Vector, Recta y Segmento

a. Desarrollar una **clase Punto** cuyos datos miembro sean las coordenadas de un punto del plano. Estos datos han de ser privados. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Punto que recibe como argumentos dos números reales, a y b y construye un nuevo objeto de la clase Punto cuyas coordenadas son a y b.
- Los métodos de acceso coordenadaX y coordenadaY, sin argumentos, que devuelven las coordenadas de un objeto Punto.
- Los métodos modificadores coordenadaX y coordenadaY, que reciben un argumento y modifican el valor de la correspondiente coordenada de un objeto Punto.
- El método igual, que comprueba si un objeto de la clase Punto es igual a otro dado que se pasa como argumento.
- El método distancia, sin argumentos, que calcula la distancia de un objeto de la clase Punto al origen de coordenadas.

**Nota:** Para calcular la distancia entre dos puntos, utilizar el teorema de Pitágoras calculando su hipotenusa (longitud de la recta que une los dos puntos)


b. Desarrollar una **clase Vector** para representar vectores. Los datos miembro de esta clase son las coordenadas del vector. Estos datos han de ser privados. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Vector que recibe como argumentos dos números reales, a y b y construye un nuevo objeto de la clase Vector cuyas coordenadas son a y b.  

$$\vec{u} = \langle a, b \rangle$$
- El constructor Vector que recibe como argumento un objeto de la clase Punto y construye un nuevo objeto de la clase Vector cuyas coordenadas coinciden con las del objeto de la clase Punto.
- El constructor Vector que recibe como argumentos dos objetos de la clase Punto, P1 y P2, y construye un nuevo objeto de la clase Vector que representa el vector de origen P1=(a<sub>1</sub>,b<sub>1</sub>) y extremo P2=(a<sub>2</sub>,b<sub>2</sub>).

$$\vec{u} = \langle a_2 - a_1, b_2 - b_1 \rangle$$



 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			 AENOR <b>R</b> Empresa Registrada ER-0934 / 2008 CERTIFIED <b>ISO 9001</b> QUALITY SYSTEM
<b>FP Parcial Presencial</b>	Pág. 9 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

- d) Los métodos de acceso extremoX y extremoY, sin argumentos, que devuelven las coordenadas de un objeto Vector.
- e) Los métodos modificadores extremoX y extremoY, que reciben un argumento y modifican el valor de la correspondiente coordenada de un objeto Vector.
- f) El método igual, que comprueba si un objeto de la clase Vector es igual a otro dado que se pasa como argumento.

$$a_1 = a_2 \text{ y } b_1 = b_2$$

- g) El método longitud, sin argumentos, que calcula la longitud de un objeto de la clase Vector.

$$\|\vec{u}\| = \sqrt{a^2 + b^2}$$

- h) El método proporcional, que comprueba si un objeto de la clase Vector es proporcional a otro dado que se pasa como argumento.

$$a_1:a_2=b_1:b_2$$

- i) El método perpendicular, que comprueba si un objeto de la clase Vector es perpendicular a otro dado que se pasa como argumento.

$$a_1 \bullet a_2 + b_1 \bullet b_2 = 0$$

- j) El método traslada, que recibe como argumento un objeto de la clase Punto, P, y devuelve el objeto Punto resultado de trasladar el punto P(X,Y) usando un objeto de la clase Vector.

$$P + \vec{u} = \langle X+a, Y+b \rangle$$

- c. Desarrollar una clase **Recta** para representar líneas rectas. Los datos miembro de esta clase son un objeto de la clase Punto perteneciente a la recta y un objeto de la clase Vector que representa la dirección de la recta. Estos datos han de ser privados. Para esta clase se piden los siguientes constructores y métodos:

- a) El constructor Recta que recibe como argumentos un objeto de la clase Punto, P, y un objeto de la clase Vector, v, y construye un nuevo objeto de la clase Recta que representa a la recta que pasa por P con dirección v.
- b) El constructor Recta que recibe como argumento un objeto de la clase Vector, v, y construye un nuevo objeto de la clase Recta que representa a la recta que pasa por el origen de coordenadas con dirección v.
- c) El constructor Recta que recibe como argumentos dos objetos de la clase Punto, P1 y P2, y construye un nuevo objeto de la clase Recta que representa a la recta que pasa por P1 y P2.

 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			 AENOR <b>R</b> Empresa Registrada ER-0934 / 2008
<b>FP Parcial Presencial</b>	Pág. 10 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

- d) Los métodos de acceso punto y vector, sin argumentos, que devuelven respectivamente los objetos Punto y Vector, datos miembro de un objeto Recta.
  - e) Los métodos modificadores punto y vector, que respectivamente reciben como argumento un objeto de la clase Punto y un objeto de la clase Vector, y modifican el correspondiente dato miembro de un objeto Recta.
  - f) El método igual, que comprueba si un objeto de la clase Recta es igual a otro dado que se pasa como argumento.
  - g) El método perpendicular, que comprueba si un objeto de la clase Recta es perpendicular a otro dado que se pasa como argumento.
  - h) El método paralela, que comprueba si un objeto Recta es paralelo a otro que se pasa como argumento.
  - i) El método pertenece, que recibe como argumento un objeto Punto P, y comprueba si P se encuentra en la recta representado por un objeto Recta.
  - j) El método paralelaPunto, que recibe como argumento un objeto Punto P, y construye la representación de la recta paralela a un objeto Recta que pasa por el punto P.
  - k) El método perpendicularPunto, que recibe como argumento un objeto Punto P, y construye la representación de la recta perpendicular a un objeto Recta que pasa por el punto P.
- d. Desarrollar una clase Segmento para representar segmentos. Los datos miembro de esta clase son dos objetos de la clase Punto extremos de un segmento. Estos datos han de ser privados. Para esta clase se piden los siguientes constructores y métodos:
- a) El constructor Segmento que recibe como argumentos dos objetos de la clase Punto, P1 y P2, y construye un nuevo objeto de la clase Segmento cuyos extremos son P1 y P2.
  - b) Los métodos de acceso extremoA y extremoB, sin argumentos, que devuelven los extremos de un objeto Segmento.
  - c) Los métodos modificadores extremoA y extremoB, que reciben como argumento un objeto de la clase Punto y modifican el correspondiente extremo de un objeto Segmento.
  - d) El método igual, que comprueba si un objeto de la clase Segmento es igual a otro dado que se pasa como argumento.
  - e) El método longitud, sin argumentos, que calcula la longitud de un objeto de la clase Segmento.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

 <b>IES Polígono Sur</b> c/ Esclava del Señor,2 41013. Sevilla	<b>PROGRAMACIÓN</b>			 AENOR <b>E</b> Empresa Registrada ER-0934 / 2008 CERTIFIED <b>IFSNet</b> QUALITY SYSTEM
<b>FP Parcial Presencial</b>	Pág. 11 de 11	<b>UNIDAD DIDÁCTICA 4 – R2- Proyectos Java</b>	PRO	Curso 2011/12

- f) El método proporcional, que comprueba si un objeto de la clase Segmento es proporcional a otro dado que se pasa como argumento.
- g) El método pendiente de la recta entre dos puntos P1=(a<sub>1</sub>,b<sub>1</sub>) y P2=(a<sub>2</sub>,b<sub>2</sub>).

$$\frac{b_2 - b_1}{a_2 - a_1}$$

- h) El método perpendicular, que comprueba si un objeto de la clase Segmento es perpendicular a otro dado que se pasa como argumento.

$$\text{Pendiente}(S_1) \times \text{Pendiente}(S_2) = -1$$

- i) El método paralelo, que comprueba si un objeto Segmento es paralelo a otro que se pasa como argumento.

$$\text{Pendiente}(S_1) = \text{Pendiente}(S_2)$$

- j) El método puntoMedio, sin argumentos, que devuelve el objeto Punto que representa el punto medio de un objeto Segmento.

$$\left( \frac{a_2 - a_1}{2}, \frac{b_2 - b_1}{2} \right)$$

- j) El método pertenece, que recibe como argumento un objeto Punto P, y comprueba si P se encuentra en el segmento representado por un objeto Segmento.

Un punto Q(Q<sub>x</sub>,Q<sub>y</sub>) pertenece a un segmento si:

- $P_{1x} < Q_x < P_{2x} \ \&\& \ P_{1y} < Q_y < P_{2y}$
- El vector  $\vec{P_1Q}$  es proporcional a  $\vec{P_1P_2}$

- k) El método recta, sin argumentos, que construye un objeto Recta que contiene a un objeto Segmento.
- l) El método mediatriz, sin argumentos, que devuelve un objeto Recta que representa la mediatriz de un objeto Segmento.

Mediatriz: Perpendicular por el punto medio.