Practica 1. Distancia.

1. Objetivos

- Realizar pequeños programas en Java, poniendo en práctica los conceptos estudiados en los primeros temas del módulo.
- Realizar programas independizando el interface de los cálculos que se hacen.
- Desarrollar clases con métodos de utilidad.
- Practicar la programación modular.
- Implementar programas en los que intervienen varias clases.

2. Introducción

Vamos a desarrollar un programa que calcule la distancia entre dos puntos de la tierra, dadas su coordenadas. El programa solicitará al usuario las coordenadas de los dos puntos y mostrará la distancia entre ambos.

Para ello se desarrollarán varias clases;

- una con el interfaz de usuario, es decir, el método main, que solicitará la información al usuario y mostrará los resultados.
- otra con los métodos necesarios para realizar los cálculos que permiten determinar la distancia.

En la práctica, aprenderemos también a desarrollar programas que tienen su código repartido en varias clases (varios archivos .java). Además, las clases desarrolladas estarán en proyectos distintos, por lo que necesitaremos saber cómo se indica, en Eclipse, que un proyecto hace uso de clases que se encuentran en otro proyecto.

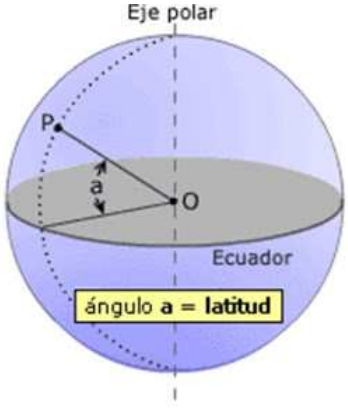
3. El problema.

Los sistemas de posicionamiento GPS proporcionan la posición de un punto utilizando tres datos: la latitud, la longitud y la altitud.

Latitud, longitud y altitud

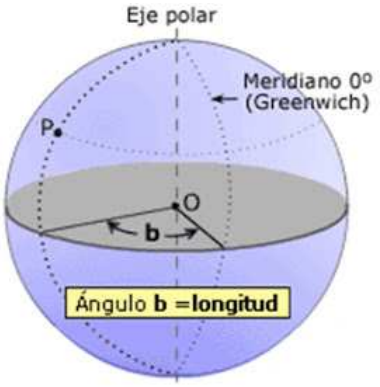
Para dar las coordenadas de un punto **sobre la superficie** terrestre se utiliza la latitud y la longitud.

La **latitud** de un punto es el ángulo que existe desde ese punto de la Tierra con respecto al Ecuador



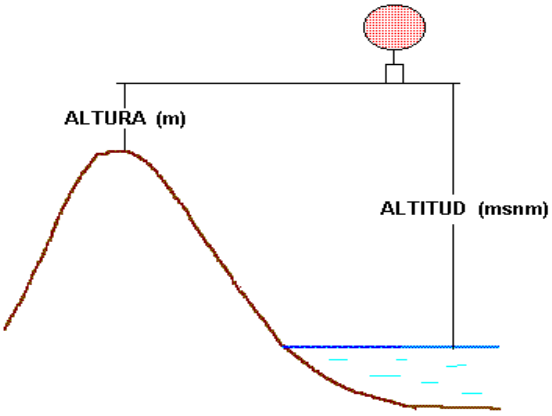
- Latitud de un punto P.

La **longitud** de un punto es el ángulo que existe desde ese punto de la Tierra con respecto al meridiano de Greenwich.



- Longitud de un punto P.

La **altitud** es la distancia vertical de un punto con respecto a un origen. Como origen suele tomarse el nivel medio del mar.



Unidades de medida.

La altitud se mide, evidentemente, en **metros**.

La latitud y la longitud, como se extrae de las definiciones de latitud y longitud, se mide en **grados,** puesto que se trata de ángulos. Los ángulos se pueden representar con diversos sistemas de medida, pero lo habitual es medirlos con **grados sexagesimales**:

Un grado sexagesimal esta definido partiendo de que una circunferencia se compone de 360° y un ángulo recto tiene, por tanto, 90°

Notación sexagesimal

Las divisiones del ángulo son el minuto sexagesimal y el segundo sexagesimal, de la siguiente forma:

- Un angulo recto = 90° (grados sexagesimales)

- Un grado sexagesimal = 60' (minutos sexagesimales)

- Un minuto sexagesimal = 60” (segundos sexagesimales).

Las partes inferiores al segundo se expresan como parte decimal de segundo.

Ejemplos

12°34’34”

13°3’23,8”

124°45’34,70”

-2°34’10”

Notación decimal

Los ángulos sexagesimales también se pueden expresar indicando únicamente los grados con decimales, teniendo en cuenta que un 1' = 1°/60 y que 1'' = 1'/60.

Así, por ejemplo, 12°15’23” = 12° + 15(1/60)° + 23,5(1/3600)° ≈ 12,25652778°

Los sistemas GPS suelen proporcionar las coordenadas en notación sexagesimal, pero para realizar ciertos cálculos resulta más cómodo utilizar la notación decimal

Distancia entre dos puntos sobre la superficie terrestre.

Para calcular la distancia entre dos puntos sobre la superficie terrestre se puede utilizar la fórmula de Harversine-

La formula Harvesine para la aproximación esférica de la distancia (d) entre dos puntos de la superficie terrestre se encuentra a continuación:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

donde Φ1, Φ2 y λ1, λ2 se refieren a la latitud y a la longitud, **expresadas ambas en radianes**, de los puntos 1 y 2 respectivamente y r corresponde al radio terrestre (6371.0 km de media).

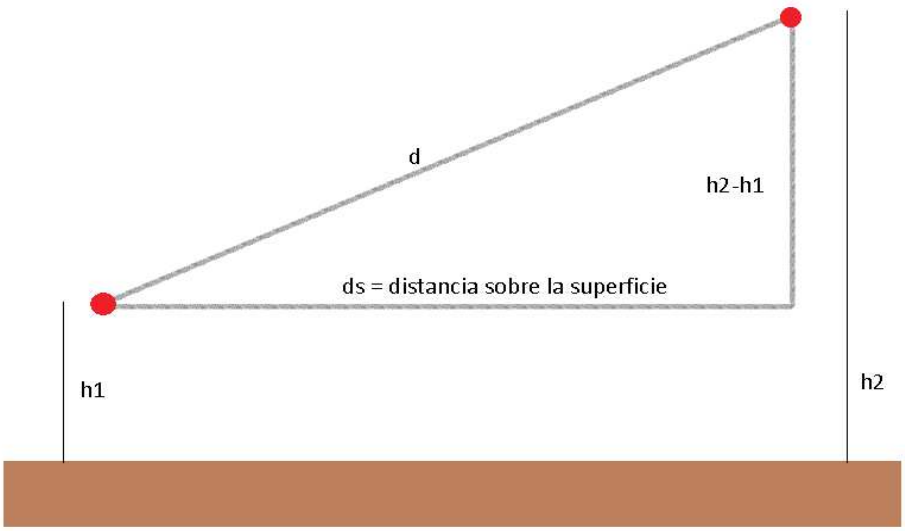
Distancia entre dos puntos teniendo en cuenta la altitud.

Conocida la distancia entre dos puntos sobre la superficie terrestre, es sencillo aproximar la distancia real entre los puntos teniendo en cuenta sus altitudes. Para ello se utiliza el teorema de Pitagoras, donde:

- La hipotenusa es la distancia que pretendemos calcular.

- Uno de los catetos es la distancia sobre la superficie terrestre.

- El otro cateto es la diferencia de las altitudes de los puntos (en valor absoluto).



4. La solución.

La solución a la práctica constará de dos proyectos:

- pro***XXX***.practica1Distancia: Contendrá un programa que pide al usuario las coordenadas de dos puntos y le muestra la distancia entre ambos.
- pro***XXX.***utiles: Este proyecto contendrá clases con métodos de utilidad que iremos desarrollando en esta práctica y en las sucesivas. Las clases y métodos contenidas en este proyecto podrían resultar útiles en otros proyectos.

(donde XXX será tu primer apellido y nombre, por ejemplo proGarciaJavier.utiles)

El proyecto proXXX.utiles

Abre el entorno de desarrollo y sigue los siguientes pasos:

1. Crea el proyecto *pro.utiles*
2. Crea en él el paquete *coordenadas*
3. Crea en él la clase Coordenadas

El proyecto proXXX.practica1Distancia

1. Crea el proyecto *pro.practica1Distancia*
2. Crea en él el paquete *distancia*
3. Crea en él la clase Distancia

Relacionar dos proyectos

Desde el proyecto practica1Distancia vamos a usar la clase Coordenadas, que se encuentra en el proyecto utiles. Para hacer ésto posible hay que indicar a Eclipse que vamos a utilizar clases que se encuentran en otro proyecto. Para hacerlo, sigue los pasos que se indican en el Anexo II, al final del enunciado.

La clase Coordenadas

Esta clase contendrá métodos útiles para nuestro propósito, calcular la distancia entre dos puntos. Se desarrollarán, al menos, los siguientes métodos, todos ellos *public static*

1. Método que convierte de grados de notación sexagesimal (grados, minutos y segundos) a notación decimal (grados)
2. Método que calcula la distancia entre dos coordenadas terrestres, según la fórmula de Harversine. Es decir, sin tener en cuenta la altitud de los puntos, sino sólo sus longitudes y latitudes. Las coordenadas vendrán expresadas en grados decimales. La fórmula de Harversine requiere que los grados estén expresados en Radianes. La clase Math dispone de un método para convertir a radianes.
3. Método que calcula la distancia entre dos puntos,dadas su altitud y la distancia que los separa en la horizontal (Pitágoras). Las coordenadas vendrán expresadas en grados decimales.
4. Método que calcula la distancia entre dos puntos dada su longitud, latitud y altitud. Este método se apoyará en los de los apartados 2 y 3 anteriores. Las coordenadas vendrán expresadas en grados decimales.

La clase Distancia.

Esta clase es la que contiene el método main. Solicitará al usuario los datos necesarios, realizará los cálculos usando los métodos de la clase Coordenadas y mostrará los resultados de nuevo al usuario:

1. Solicitará al usuario las coordenadas de dos puntos. Para cada punto:
    - latitud (horas, minutos y segundos)
    - longitud (horas minutos y segundos)
    - altitud

    Es decir, 14 datos en total
2. Apoyándose en la clase Coordenadas, realizará los cálculos necesarios para mostrar la información que se indica a continuación.
3. Mostrará por pantalla la siguiente información:
    - Distancia, en metros, entre los puntos sin tener en cuenta sus altitudes.
    - Distancia, en metros, entre los puntos teniendo en cuenta sus altitudes.
    - Diferencia de latitud, en metros, entre los puntos.

5. Evaluación de la práctica

Se valorará:

- Que los proyectos estén organizados tal y como propone el enunciado (proyectos, paquetes, clases y métodos).

- Que los proyectos se nombren tal y como indica el enunciado.
- Que las identificadores (de métodos, variables y clases) sean adecuados: nombres significativos y siguiendo el criterio de mayúsculas y minúsculas recomendado por Java.
- La correcta identación del código y su legibilidad.
- La definición de constantes cuando se de el caso.
- La elección del **tipo de las variables**: En especial la distinción de datos enteros y reales y el consumo mínimo de memoria de forma que, por ejemplo, si para almacenar un dato es suficiente con un byte, no se utilice un short, si es suficiente con un short, no se utilice un int, y así sucesivamente.

6. Entrega de la práctica

1. Localiza la carpeta de proyecto en tu workspace y comprímela en un archivo y renómbralo como XXXpractica1 (y la extensión correspondiente), donde XXX será tu apellido y nombre.
2. Súbelo al aula virtual: Junto a la carpeta de la práctica encontrarás un apartado que te permitirá subir el archivo comprimido.
3. Plazo entrega: Hasta el martes 27 de octubre a las 23:55.

7. Anexo I: Programas compuestos por varias clases.

Para invocar a un método desde otro método, basta con escribir el nombre del método al que se llama y pasarle los parámetros necesarios. Sin embargo, si el método que hace la llamada y el método al que se llama se encuentran en clases distintas, esto no basta.

Si queremos invocar a un método que se encuentra en otra clase utilizamos la siguiente sintáxis:

NombreDeClase.nombreDelMétodo(parámetros)

Por ejemplo,

para llamar al método *round* de la clase *Math*

```
... Math.round(numero)
```

8. Anexo II: Utilizar clases que se encuentran en otro proyecto

Cuando en un proyecto queremos utilizar clases que se encuentran en otro proyecto, hay que indicarlo convenientemente a Eclipse:

Si desde el proyecto ProyA queremos usar clases que se encuentran en el proyecto ProyB, tendremos que seguir los siguientes pasos:

1. Haz clic con el botón derecho en ProyA.

2. Selecciona Build Path – Configure Build Path.

3. En la ventana que aparece, selecciona la pestaña *Projects* y pulsa el botón *Add.*

4. En la ventana que aparece, selecciona el proyecto requerido, es decir, ProyB y pulsa OK.