**Generalidades**

1. Crear un proyecto de java con Maven con artifact id “homework” y group id “ec.edu.espol”.
2. Su proyecto solo debe tener una clase Main, las actividades que solicitan en el Main deben ser desarrolladas en esta clase.
3. Crear el paquete ec.espol.edu.model donde debe crear las siguientes clases:

**Ejercicio 1**

1. Crear la clase Vector2D
   1. Los atributos de la clase vector son la posición **x** y **y** del vector como double.
   2. Sobrecargar el constructor para que solo reciba la posición x. La posición **y** tendrá un valor de 0.
   3. Sobrecargar el constructor para que reciba el valor de **x** y **y**.
   4. Implementar getters y setters.
   5. Implementar los métodos **sumar**, y **restar** entre dos vectores. Estos métodos dan como resultado un nuevo vector.
   6. Implementar el método módulo que calcula el módulo del vector. Usar Math.sqrt(x) para calcular la raíz cuadrada. Usar Math.pow(base, exponente) para calcular una potencia.
   7. Implementar el método vector unitario que retorna un nuevo vector que representa el vector unitario.
   8. Implementar el toString para que se convierta a String con el siguiente formato: **v = (3,5). (El formato incluye la v y el igual)**
   9. Implementar el método equals que permite comparar si dos vectores son igual. Dos vectores son iguales si tienen el mismo valor de **x** y **y**.
   10. Crear un método estático que reciba un objeto Scanner y proceda a crear un vector a partir de la información que recibe por teclado (posición **x** y **y**).
2. En la clase Main recibir dos vectores por teclado, validar que sean valores positivos los valores de **x** y **y** (incluido el 0). Luego deberá presentar los resultados de las operaciones sumar, y restar entre estos dos vectores. Finalmente presentar un mensaje si ambos vectores son iguales.

**Ejercicio 2**

1. Crear la clase Array
   1. El atributo de la clase es un arreglo de enteros.
   2. Sobrecargar el constructor para que reciba el tamaño del array. El constructor procede a inicializar el arreglo con el tamaño recibido.
   3. Declarar el constructor por default. Este constructor inicializa el arreglo con una dimensión de 10.
   4. Implementar el método size que retorna un entero que representa el tamaño del arreglo.
   5. Implementar el método concat que recibe un objeto Array. El método procede a generar un nuevo objeto Array que va a tener los elementos del this y los elementos del objeto array que recibió. Este método es similar a lo que hace el operador + con dos listas, retorna una nueva lista con los elementos de ambas listas.
   6. Implementar el método get que recibe un índice y procede a retornar el entero que está en el índice correspondiente. En caso de que el usuario envíe un índice que no corresponde a los índices válidos, retornar -1.
   7. Implementar el método set que recibe un índice y un valor entero. El método procede a reemplazar el valor entero en el índice que recibe como argumento. En caso de que el usuario envíe un índice que no corresponde al rango de índices, no se procede a realizar el cambio.
   8. Implementar el método slicing que recibe dos índices (inicio y final). El método retorna un nuevo objeto array con los elementos que se encuentran en el intervalo proporcionado. El slicing incluye el inicio, pero no incluye el final. En caso de que los índices recibidos como argumentos no corresponda a un intervalo válido, retornar null.
   9. Implementar el método sum que retorna la suma de los elementos del arreglo.
   10. Implementar el método max que retorna el valor máximo de los elementos del arreglo.
   11. Implementar el método min que retorna el valor mínimo de los elementos del arreglo.
   12. Implementar el toString para que se convierta a String con el siguiente formato: **[2 3 4 5]**
   13. Implementar el método equals. Hay que considerar que dos Array de diferentes dimensiones no son iguales. Para que sean iguales deben tener los mismos elementos y en el mismo orden.
   14. Crear un método estático que reciba un objeto Scanner y proceda a crear un Array a partir de la información que recibe por teclado. Se solicita la dimensión del Array y a partir de esa información se solicitan los valores del Arreglo.
   15. En la clase Main pedir por teclado 2 instancias de Array y probar todas las operaciones solicitadas en este ejercicio.

**NOTA.- índices inválidos se refiere a índices negativos o que estén fuera del rango de índices del arreglo. En java no existen los índices negativos.**

**Ejercicio 3**

Crear la clase **Mail** que contiene los siguientes atributos:

* Correo origen
* Correo destino
* Titulo
* Mensaje

1. Crear un **constructor** que reciba todos los valores para inicializar los atributos de la clase.
2. Crear **getters** y **getters**.
3. Sobrescribir el método **toString**. El formato que debe tener es el siguiente: **<from:ruben@org.com, to:maria@org.com, tittle: correo de prueba, message: esto es un mensaje de prueba>**
4. Sobrescribir el método **equals**, dos mails son iguales si tienen el mismo correo de origen, correo de destino, título, y mensaje.
5. Crear el comportamiento **nextMail** que recibe un objeto Scanner y procede a retornar un objeto Mail con valores obtenidos por teclado.

Crear la clase Buzon que contiene los siguientes atributos:

* Una lista de objetos Mails.

1. Crear el constructor default de la clase Buzon que no recibe nada y procede a inicializar la lista de objetos Mails vacía.
2. Crear **getters**.
3. Sobrescribir el método **toString**. El método toString debe presentar los Mails que tiene el Buzón con el siguiente formato:

**{ Mail #1: <from:ruben@org.com, to:maria@org.com, tittle: correo de prueba, message: esto es un mensaje de prueba>, Mail #2: <from:carla@org.com, to:julio@org.com, tittle: correo de prueba, message: esto es un mensaje de prueba>, Mail #3: <from:luis@org.com, to:maria@org.com, tittle: correo de prueba, message: esto es un mensaje de prueba>}**

1. Implementar un comportamiento **addMail** que recibe un objeto Mail y procede a agregarlo a la lista de Mails del buzón.
2. Implementar el comportamiento **filtrarBuzonfromSrc** que recibe un String con un correo origen y retorna una nueva lista de objetos Mails que contiene aquellos Mails del buzón cuyo correo de origen es igual al correo recibido en el argumento de la función.
3. Implementar el comportamiento **filtrarBuzonfromDst** que recibe un String con un correo destino y retorna una nueva lista de objetos Mails que contiene aquellos Mails del buzón cuyo correo de destino es igual al correo recibido en el argumento de la función.

En la clase Main

* Crear un objeto Buzon.
* Solicitar 5 objetos Mails por teclado y agregarlos al objeto buzón.
* Presentar todos los Mails que tiene el buzón.
* Solicitar un correo origen por teclado, filtrar los Mails del buzón por este correo origen y presentar por pantalla la información de los Mails filtrados.
* Solicitar un correo destino por teclado, filtrar los Mails del buzón por este correo destino y presentar por pantalla la información de los Mails filtrados.