Unidad 1: Introducción a la Orientación a Objetos
Concepto de clase, instancia, atributo de instancia, método de instancia, atributo de clase y método de clase. Diferencia entre atributo de instancia y de clase. Diferencia entre método de instancia y método de clase. Constructores.

Ejercicio 1
Diseñe e implemente la clase **BankAccount**, la misma debe contar con los siguientes métodos:

- *deposit*(float amount) : deposita una cantidad de dinero que se debe comprobar que sea positiva.
- *withdraw*(float amount): retira una cantidad de dinero que se debe comprobar que sea positiva.
- *getBalace*(): que devuelva el balance de la cuenta.
- *getCBU*(): devuelve el CBU de la cuenta.

Las cuentas se identifican con un CBU.

Ejercicio 2
Diseñe e implemente la clase **Rectangle**, definida por los atributos de *base* y *height* como números enteros. Defina el método calculateArea(), el cual calcula el área de un rectángulo a partir de su base y altura. Luego:

- Cree el objeto Rectangle1.
- Instancie el atributo *base*, del objeto Rectangle1, con el valor 10.
- Instancie el atributo *height*, del objeto Rectangle1, con el valor 5.
- Calcule el área del Rectangle1 y muestre el resultado por pantalla.
- Redefina la clase Rectangle añadiendo el constructor Rectangle(int b, int h)
- Cree el objeto Rectangle2 (base = 23 y altura = 7), calcule su área y muestre el resultado por pantalla.

Ejercicio 3
Diseñe e implemente la clase **Point2D**, la cual debe representar un punto en el espacio cartesiano de dos dimensiones. Debe contar con los siguientes métodos:

- getDistance(Point2D point): que indique la distancia que existe entre el punto destino.
- add(Point2D point): que devuelva un nuevo punto la suma de el punto original y el argumento.
- getX() : devuelve la coordenada en X
- getY(): devuelve la coordenada en Y

Defina el método de clase getDistanceToOrigin(Point2D point) que devuelve la distancia entre un punto y el origen de coordenadas (0,0).
Defina el atributo de clase *maxDistanceToOrigin,* el cual indica la máxima distancia que existe entre el origen de coordenadas y cualquier punto creado a partir de la clase Point2D.
Finalmente, cree 3 objetos diferentes de la clase Point2D, con diferentes valores para las coordenadas cartesianas x y. Muestre por pantalla cual es la máxima distancia calculada al origen de coordenadas, a partir de los 3 puntos creados previamente.

Puede utilizar la funciones: (más adelante veremos que son)

- Math.pow(x, y) : eleva el número x a la potencia y.
- Math.sqrt(x): devuelve la raíz cuadrada de x.

Ejercicio 4

Implemente la clase círculo, que esté definida cómo un centro y un rádio. De tal manera que permita consultar su centro, su área e indicar su un punto está incluído en su área:

- center() : obtiene el centro del círculo
- area(): obtiene el área del círculo
- contains(Point2D point): devuelve True/False si el punto está contenido está contenido.
- perimeter(): devuelve el perímetro del círculo.

Puede utilizar la constante π haciendo uso de: Math.PI

Ejercicio 5

La recepción de una empresa, habilitada para ejercer sus actividades en esta cuarentena, desea controlar el ingreso de todas las personas a la institución. Para cumplir con esta actividad la empresa desea adquirir un sistema para dicho control. Dentro del proyecto de desarrollo del sistema de control de ingreso para la empresa, se le solicita que modele e implemente una clase para registrar a las personas que ingresan a la institución. La clase debe poder registrar el nombre, apellido, DNI, fecha y hora de ingreso a la empresa. Además se desea constatar si la persona ingresa en calidad de empleada o visitante. Por último, se requiere saber cuántas personas fueron registradas por ingresar.
Para evaluar el desempeño de la clase implementada simule el ingreso de 3 personas a la institución. Todos los datos deben ser cargados por teclado. Finalizados los ingresos, debe mostrar por pantalla: la cantidad de personas que ingresaron y el nombre y apellido de cada una.

Ejercicio 6

Implemente la clase **Fraction** que representa una fracción, o número racional. Debe almacenar siempre la expresión más sencilla posible. Debe contar con los siguientes métodos:

- add(Fraction f): suma una fracción y obtiene una nueva.
- sub(Fraction f): resta una fracción y obtiene una nueva.
- add(int number): suma un número entero a la fracción y obtiene una nueva.
- mul(Fraction f): multiplica una fracción por otra y obtiene una nueva.
- div(Fraction f): divide una fracción por otra y obtiene una nueva.
- asFloat() : devuelve su valor cómo punto flotante.
- equals(Fraction f): indica si una fracción es igual a la otra.

Ayuda: Una implementación recursiva del MCM (máximo común múltiplo):

```
public int gcd(int a, int b) {
if (b==0) return a;
return gcd(b,a%b);
```

```
}
```