Construyendo Objetos

1. Modelar e implementar la clase **Nota** para cumplir con la siguiente interfaz:

```
class Nota {

	/* pre : valorInicial está comprendido entre 0 y 10.
	 * post: inicializa la Nota con el valor indicado.
	 */
	Nota(int valorInicial) {

	}

	/* post: devuelve el valor numérico de la Nota,
	 *       comprendido entre 0 y 10.
	 */
	int obtenerValor() {

	}

	/* post: indica si la Nota permite o no la aprobación.
	 */
	boolean aprobado() {

	}

	/* post: indica si la Nota implica la desaprobación.
	 */
	boolean desaprobado() {

	}

}
```

2. Agregar a la clase Nota el método:

```
/* pre : nuevoValor está comprendido entre 0 y 10.
 * post: modifica el valor numérico de la Nota, cambiándolo
 *       por nuevoValor, si nuevoValor es superior al
 *       valor numérico actual de la Nota.
 */
void recuperar(int nuevoValor) {

}
```

3. Modelar e implementar la clase Punto. Un Punto en el plano posee coordenada X y coordenada Y. Proporcionar métodos para:
   - consultar y cambiar las coordenadas,
   - saber si el punto está sobre el eje de las X,
   - saber si el punto está sobre el eje de las Y,
   - saber si el punto es el origen de coordenadas.

4. Implementar la clase Cubo a partir de la siguiente interfaz:

```
public class Cubo {

	/* post: inicializa el cubo a partir de la medida de lado dada
	 */
	public Cubo (int lado) {

	}
	/* post: devuelve la longitud de todos los lados del cubo
	 */
	public int medirLongitudLado() {
```

```
    }

    /* pre: lado es un valor mayor a 0.
     * post: cambia la longitud de todos los lados del cubo
     */
    public void cambiarLongitudLado(int lado) {

    }

    /* post: devuelve la superficie ocupada por las caras del cubo
     */
    public int medirSuperficieCara() {

    }

    /* pre: superficieCara es un valor mayor a 0.
     * post: cambia la superficie de las caras del cubo
     */
    public void cambiarSuperficieCara(int superficieCara) {

    }

    /* post: devuelve el volumen que encierra el cubo
     */
    public int medirVolumen() {

    }

    /* pre: volumen es un valor mayor a 0.
     * post: cambia el volumen del cubo
     */
    public void cambiarVolumen(int volumen) {

    }
}
```

5. Implementar la clase TarjetaBaja a partir de la siguiente declaración:

```
public class TarjetaBaja {

    /* post: saldo de la Tarjeta en saldoInicial.
     */
    public TarjetaBaja(double saldoInicial)

    public double obtenerSaldo()

    /* post: agrega el monto al saldo de la Tarjeta.
     */
    public void cargar(double monto)

    /* pre : saldo suficiente.
     * post: utiliza 1.25 del saldo para un viaje en colectivo.
     */
    public void pagarViajeEnColectivo()

    /* pre : saldo suficiente.
     * post: utiliza 2.50 del saldo para un viaje en subte.
     */
    public void pagarViajeEnSubte()

    /* post: devuelve la cantidad de viajes realizados.
     */
    public int contarViajes()
```

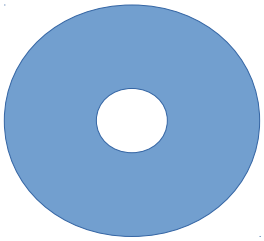
```
    /* post: devuelve la cantidad de viajes en colectivo.
     */
    public int contarViajesEnColectivo()

    /* post: devuelve la cantidad de viajes en subte.
     */
    public int contarViajesEnSubte()

}
```

6. Implementar una clase que modele una Circulo, del que se desa conocer: radio, diámetro, perímetro y superficie.

7. Implementar una clase que modele un Disco.
   Se desea conocer:



- radio interior
- radio exterior
- perímetro interior
- perímetro exterior
- superficie.

   Debe tener operaciones para cambiar el radio interior y el radio exterior.

8. Implementar la clase Ticket a partir de la siguiente interfaz

```
public class Ticket {

    /* post: el Ticket se inicializa con importe 0.
     */
    public Ticket()

    /* pre : cantidad y precio son mayores a cero. El ticket está abierto.
     * post: suma al Ticket un item a partir de la cantidad de
     *       de productos y su precio unitario.
     */
    public void agregarItem(int cantidad, double precioUnitario)

    /* pre : el Ticket está abierto y no se ha aplicado un descuento previamente.
     * post: aplica un descuento sobre el total del importe.
     */
    public void aplicarDescuento(double porcentaje)

    /* post: devuelve el importe acumulado hasta el momento sin cerrar el Ticket.
     */
    public double calcularSubtotal()

    /* post: cierra el Ticket y devuelve el importe total.
     */
    public double calcularTotal()

    /* post: devuelve la cantidad total de productos.
     */
```

```
    public int contarProductos()
}
```

9. Implementar la clase CajaDeAhorro con la siguiente interfaz:

```
public class CajaDeAhorro {

    /*
     * post: la instancia queda asignada al titular indicado
     *       y con saldo igual a 0.
     */
    public CajaDeAhorro(String titularDeLaCuenta) {

    }

    /*
     * post: devuelve el nombre del titular de la Caja de Ahorro.
     */
    public String obtenerTitular() {

    }

    /*
     * post: devuelve el saldo de la Caja de Ahorro.
     */
    public double consultarSaldo() {

    }

    /*
     * pre : monto es un valor mayor a 0.
     * post: aumenta el saldo de la Caja de Ahorro según el monto
     *       depositado.
     */
    public void depositar(double monto) {

    }

    /*
     * pre : monto es un valor mayor a 0 y menor o igual que el saldo de la
     *       Caja de Ahorro.
     * post: disminuye el saldo de la Caja de Ahorro según el monto
     *       extraido.
     */
    public void extraer(double monto) {

    }
}
```

10. Implementar la clase Cerradura con los siguientes métodos. Indique axiomas de la clase, pre y post condiciones de los métodos.

```
public class Cerradura {
        public Cerradura(int claveDeApertura, int cantidadDeFallosConsecutivosQueLaBloquean)
        public boolean abrir(int clave)
        public void cerrar()
        public boolean estaAbierta()
        public boolean estaCerrada()
        public boolean fueBloqueada()
        public int contarAperturasExitosas()
        public int contarAperturasFallidas()
}
```

Considerar que cuando una Cerradura se bloquea no puede volver a abrirse nunca más.

11. Implementar la clase ExpendedorDePasajes con los siguientes métodos. Indique axiomas de la clase, pre y post condiciones de los métodos.

```
public class ExpendedorDePasajes {
        public ExpendedorDePasajes(double precioPorKm)

        public double venderPasaje(double distanciaEnKm)

        public double venderPasajes(int cantidad, double distanciaEnKm)

        public int pasajesVendidos()

        public double distanciaMaxima()

        public double distanciaPromedio()

        public double ventaTotal()
}
```

12. Un **Tragamonedas** está compuesto por 3 **Tambores.** Cuando el **Tragamonedas** se activa, giran los 3 **Tambores**. Cada **Tambor** se detiene en una posición comprendida entre 1 y 8. El **Tragamonedas** entrega un premio cada vez que, luego de ser activado los 3 **Tambores** se detienen en la misma posición.
    Implementar la clase **Tragamonedas** y **Tambor** a partir de las siguientes interfaces:

```
public class Tambor {

        /* post: devuelve el número de posición en la que se
         *       encuentra el Tambor. Es un valor comprendido
         *       entre 1 y 8.
         */
        public int obtenerPosicion() {

        }

        /* post: hace girar el tambor y luego se detiene en
         *       una posición comprendida entre 1 y 8.
         */
        public void girar() {

        }
}

public class Tragamonedas {

        /* post: los 3 Tambores del Tragamonedas estÃ¡n
         *       en la posición 1.
         */
        public Tragamonedas() {

        }

        /* post: activa el Tragamonedas haciendo girar
         *       sus 3 Tambores.
         */
        public void activar() {

        }

        /* post: indica si el Tragamonedas entrega un premio
         *       a partir de la posición de sus 3 Tambores.
         */
        public boolean entregaPremio() {

        }
```

```
}
```

13. Implementar la clase Alarma y la clase Sensor con la siguiente interfaz. Una Alarma cuenta con un Sensor de movimiento, un Sensor de contacto y un Sensor de sonido.

```
public class Sensor {

	/*
	 * post: sensor apagado.
	 */
	public Sensor() {

	}

	/*
	 * post: enciende el sensor.
	 */
	public void encender() {

	}

	/*
	 * post: apaga el sensor.
	 */
	public void apagar() {

	}

	/*
	 * post: devuelve si el sensor ha sido activado a causa de algÃºn evento.
	 */
	public boolean activado() {

	}

	/*
	 * post: activa el sensor.
	 */
	public void activar() {

	}
}


public class Alarma {
	/*
	 * post: alarma apagada con el código de seguridad indicado.
	 */
	public Alarma(int codigoSeguridad) {

	}

	/*
	 * post: enciende la alarma.
	 */
	public void encender() {

	}

	/*
	 * post: si codigoSeguridad es correcto, apaga la alarma.
	 */
	public void apagar(int codigoSeguridad) {

	}

	/*
	 * post: devuelve si alguno de los sensores está activado.
	 */
	public boolean activada() {

	}
```

```
}
```

Para cada una de las clases construidas en todos los ejercicios, implementar una clase que constituya la prueba de la misma.