Unidad 2: **Composición**

Tipos de datos (primitivos y por referencia). Colecciones de objetos.

Ejercicio 1

Implemente la clase **Triangle**, que debe estar compuesta por Rectas, que a su vez está compuesta por dos puntos.

Debe contar con los siguientes métodos:

- area()
- perimeter()
- isIsoceles()
- isScalane()
- isEquilateral()

Indique: ¿Qué tipo de agregación tiene la clase Triángulo con sus componentes?

Ejercicio 2

Implemente la clase **BankAccount**, que cuenta con un código de cuenta, y la posibilidad de depositar y extraer, tal como consultar el balance de esta.

Implemente la clase **Bank**, el cual posee una colección de cuentas bancarias. Dicho banco debe permitir transferir dinero entre cuentas.

Indique: ¿Qué tipo de agregación tiene la clase Banco con sus componentes?

Ejercicio 3

Implemente la clase **PhoneBill**, que permite registrar movimientos de una cuenta de teléfonos (duración de la llamada). A medida que se realizan consumos, estos se van registrando y permiten luego imprimir por pantalla los movimientos, permite además consultar el saldo de la cuenta. Además, existe la posibilidad de ajustar el precio del segundo en cualquier momento.

Ejercicio 4

Una unidad de cuidados intensivos (UCI) es una instalación especial dentro del área hospitalaria que proporciona medicina intensiva. Los pacientes destinados a esta sección tienen alguna condición grave de salud que pone en riesgo su vida y por lo tanto requieren de una monitorización constante de sus signos vitales.

Diseñe e implemente un administrador de UCI, el cual debe revisar los principales monitores presentes:

- Monitor cardiaco: muestra continuamente la frecuencia cardíaca.
- Monitor respiratorio: muestra continuamente la frecuencia respiratoria.
- Monitor de presión arterial: muestra continuamente la presión arterial.
- Monitor de oxígeno transcutáneo: muestra continuamente la cantidad de oxígeno presente en la piel.

Todos estos monitores encienden una alarma si los valores medidos están fuera del rango normal.

Ejercicio 5

Java, para poder interactuar con mayor cantidad de objetos, incluyó en el lenguaje (además de los tipos básicos, boolean, byte, char, int, float y double) sus correspondiente clases, Boolean, Byte, Character, Integer, Float, Double.

1. Investigue que es el Inboxing y el Outboxing, para que estos objetos se comporten de manera indistiguida con su correspondiente tipo built-in.
2. Conteste: ¿Porqué es importante que existan estos objetos?

Ejercicio 6

En Java, el operador == compara únicamente posiciones de memoria. Es decir, que sólo puede comparar correctamente int, char, boolean, null. Los objetos que se quieran comparar, deben implementar su propio boolean equals(Object o). Sin embargo, para algunos casos, por un tema de eficiencia, Java, no crea objetos nuevos, sino que los reutiliza. Por ejemplo, dos Strings creados con la misma cadena.
Responda y verifique mediante tests unitarios, cuál es la salida de cada línea:

```
public class Equals {

   public static void main(String[] args) {
       String str1 = "Hola Mundo!";
       String str2 = "Hola Mundo!";
       String str3 = "Hola ";
       String str4 = "Mundo!";
       String str5 = str3 + str4;
       int num1 = 1;
       Integer num1_2 = 1;
       Integer num200 = 200;
       Integer num200_1 = 200;
       System.out.println(str1 == str2);
       System.out.println(str1 == str5);
       System.out.println(str1.equals(str5));
       System.out.println(num1.equals(num1_2));
       System.out.println(num1_2.equals(num1));
       System.out.println(num200 == num200_1);
       System.out.println(num200.equals(num200_1));

   }
}
```