Problemario correspondiente a la asignatura "Programación Orientada a Objetos"

Dr. José Luis López Martínez

Septiembre-Enero 2020

Índice general

Unidad 1

Introducción a la programación orientada a objetos

Contenido de la Unidad

La asignatura “Programación orientada a objetos” se imparte como asignatura obligatoria a los estudiantes de la Licenciatura en Ingeniería de Software , en la Unidad Multidisciplinaria Tizimín, en la Facultad de Matemáticas. Este Problemario se encuentra elaborado con base en los contenidos de dicha asignatura.

Este Problemario presenta una relación de ejercicios para cada una de las unidades de la asignatura y para la secuencia de contenidos de cada una de ellas.

Nombre de la asignatura	Programación Orientada a Objetos				
Tipo	Obligatoria				
Modalidad	Mixta				
Ubicación	Tercer semestre				
Duración total en horas	128	Horas presenciales	72	Horas no presenciales	56
Créditos	8				
Requisitos académicos previos	Ninguno				

1.1. El Paradigma de Programación Orientado a Objetos (POO)

1. Menciona brevemente al menos cinco principales estilos o paradigmas de programación e incluye una breve descripción

2. Menciona al menos tres objetivos del paradigma orientado a objetos.

3. Define los conceptos de *Reusabilidad* y *Especialización*

4. Describe el concepto de *Abstracción*

5. Menciona los componentes de un TDA

1.2. Lenguajes Orientados a Objetos

1. Investigar las principales características de los siguientes lenguajes de programación orientados a objetos: Ada, C++, Java, Python.

2. Menciona las tres características básicas de un lenguaje orientado a objetos

3. Subraya la respuesta correcta, recuerda que estas son las características esenciales de un lenguaje orientado a objetos

    - Es una abstracción conceptual del mundo real que se puede traducir a un lenguaje computacional o de programación Orientada a Objetos. Esta se caracteriza por tener una identidad única que lo distingue de otros objetos.

   a) Abstracción.
   b) Objeto
   c) Modularidad
   d) Clase

- Es una colección de objetos similares. Estos objetos deben tener los mismos atributos con valores posiblemente diferentes asignados a estos, y el mismo conjunto de métodos que definen su comportamiento.
   a) Abstracción.
   b) Objeto
   c) Modularidad
   d) Clase
- Denota las características esenciales de un objeto que lo distinguen de todos los demás tipos de objeto y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador
   a) Abstracción
   b) Objeto
   c) Modularidad
   d) Clase
- Es una clasificación u ordenación de abstracciones
   a) Abstracción.
   b) Jerarquía
   c) Modularidad
   d) Clase
- Este principio significa que las estructuras de datos internas utilizadas en la implementación de una clase no pueden ser accesibles directamente al usuario de la clase
   a) Abstracción.
   b) Objeto
   c) Modularidad
   d) Encapsulamiento
- Es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados
   a) Abstracción.
   b) Objeto
   c) Modularidad
   d) Encapsulamiento

4. Menciona al menos tres IDES (Integrated Development Environment) para el lenguaje de programación Java y un link para su descarga.

Unidad 2

Clases y Objetos

Contenido de la Unidad

2.1. Abstracción y clasificación

1. Identifica y liste 2 *objetos* de un hospital, identifica algunos *estados* y un *comportamiento*

2. Identifica las clases necesarias para un sistema de registro de calificaciones de una escuela considerando Objetos, Estados y Comportamientos

3. Identifique algunas clases que se encuentran en un Salón de clases, considerando Objetos, Estados y Comportamientos

4. Identifique el estado y comportamiento de una clase *Cliente* de un Banco

2.2. Definición de clases y creación de objetos

1. Relaciona los modificadores de acceso con que o quien puede acceder a los atributos/métodos declarados con los mismos

Acceso	Descripción
( )Public	1-Las clases que están al mismo nivel (dentro del paquete).
( )Protected	2-Únicamente la misma clase a la que pertenece
( )Private	3-Las clases que heredan de una superclase
( )Package	4-Todo tipo de clases

2. Subraya la respuesta correcta de las opciones presentadas, algunas preguntas puede ser de tema vistos con anterioridad.

   - Si desea usar una clase existente de la Biblioteca de clases Java en su programa, ¿qué palabra clave debe usar?
     a) export
     b) use
     c) include
     d) import
   - Un constructor tiene el mismo nombre que el nombre de la clase.
     a) Verdadero
     b) Falso
   - Una clase dada puede tener más de un constructor
     a) Verdadero
     b) Falso

- En una clase determinada llamada Prueba , solo puede haber un método con el nombre Prueba
  - *a*) Verdadero
  - *b*) Falso
- Un método estático es también llamado como un
  - *a*) un método de clase
  - *b*) un método de instancia
- La clase Prueba tiene un método llamado *zoo*, la cual tiene el siguiente encabezado: *public static double zoo (float f)* , ¿Qué podemos decir acerca del método zoo?
  - *a*) un método de clase
  - *b*) un método de instancia
  - *c*) es una variable de instancia
  - *d*) es un campo de la clase
- En la clase Prueba , el método zoo tiene la siguiente API: *public static void zoo ( )* , ¿Cómo podemos llamar a ese método?
  - *a*) Prueba.zoo(8);
  - *b*) new Prueba(zoo());
  - *c*) Prueba.zoo()
  - *d*) Prueba (zoo())
- En la clase Prueba , el método zoo tiene la siguiente API: *public double zoo (int i, String s, char c)* , ¿Cuántos argumentos toma el método zoo?
  - *a*) 0
  - *b*) 1
  - *c*) 2
  - *d*) 3
- En la clase Prueba , el método zoo tiene la siguiente API: *public double foo (int i, String s, char c)* , ¿Cual es el tipo de valor que regresa el método?
  - *a*) double
  - *b*) int
  - *c*) char
  - *d*) String
- El String es un tipo de datos primitivos de Java
  - *a*) Falso
  - *b*) Verdadero
- Cual es la forma apropiada de acceder a la constante E de la clase *Math*

a) E;
b) Math(E)
c) Math.E
d) Math(E)

3. Escriba un programa que solicite al usuario un nombre de dominio. Su programa debería concatenar ese nombre con www . y .com para formar un nombre de dominio de Internet y generar el resultado. Por ejemplo, si el nombre ingresado por el usuario es google , la salida será www.google.com

4. Escriba un programa que pida al usuario un valor (double) como entrada, luego calcule y genere el cubo de ese número usando el método pow de la clase Math e imprima el resultado en pantalla. Debes de utilizar al menos dos clases.

5. Escriba un programa que pida al usuario el radio de un circulo (double) como entrada, luego calcule y obtenga el área y el perímetro del circulo. La formula del area es πr^2 y el perímetro es $2\pi r$, donde r es el radio del circulo.

2.3. Métodos

1. Subraya la respuesta correcta de las opciones presentadas, algunas preguntas puede ser de tema vistos con anterioridad.

   - ¿Qué puedes decir sobre el nombre de una clase?
     a) Debe comenzar con una letra mayúscula.
     b) La convención es comenzar con una letra mayúscula
   - ¿Qué puedes decir sobre el nombre de los constructores?
     a) Deben tener el mismo nombre que el nombre de la clase.
     b) Pueden ser cualquier nombre, al igual que otros métodos.
   - ¿Cuál es el tipo de retorno de un constructor?
     a) Objeto
     b) nulo
     c) El nombre de la clase
     d) Un constructor no tiene tipo de retorno
   - ¿Es legal tener más de un constructor en una clase dada?
     a) Verdadero
     b) Falso
   - En una clase, si un campo es privado,
     a) se puede acceder directamente solo desde dentro de su clase

b) se puede acceder directamente desde cualquier clase.

- En una clase típica, ¿cuál es la recomendación general para los modificadores de acceso?
  - *a*) Los atributos son privados y los métodos son privados
  - *b*) Los atributos son públicos y los métodos son privados.
  - *c*) Los atributos son privados y los métodos son públicos
  - *d*) Los atributos son públicos y los métodos son públicos.
- ¿Cuál es la palabra clave utilizada para declarar una constante?
  - *a*) static
  - *b*) constant
  - *c*) final

2. Del código proyectado. Identifica lo que se te pide.
   - *a*) ¿Cuál es el nombre de las clases?
   - *b*) ¿Cuál es la clase principal?
   - *c*) Escribe los atributos de alguna de las clase
   - *d*) ¿Cuál es el método constructor?
   - *e*) ¿Que atributo sería equivalente a una variable global
   - *f*) Escribe la línea de código en donde se está creando algún objeto
   - *g*) A qué atributos referencia this (escribe el numero de línea de código donde se encuentra el atributo)

Listing 2.1: Clase ApCuentaBancaria

```
1 public class ApCuentaBancaria{
2
3 public static void main(String[] args){
4    CuentaBancaria c1=new CuentaBancaria(123,"Julio",2000);
5  }
6
7 }
```

Listing 2.2: Clase CuentaBancaria

```
1 class CuentaBancaria{
2    long numero;
3    String titular;
4    long saldo;
5
6    CuentaBancaria(long num, Cliente clt, long s) {
```

```
7           numero=num;
8           titular=clt;
9           saldo=s;
10      }
11
12  void ingresar(long cantidad){
13       this.saldo += cantidad;
14      }
15
16  }
```

Listing 2.3: Clase Globales

```
1  public class Globales{
2      public static String nombreEscuela="UADY";
3   }
```

3. Realiza una clase llamada Estudiante que tenga las siguientes condiciones:

   - Sus atributos: nombre, edad, matricula, sexo (H hombre, M mujer), peso y altura. No queremos que se accedan directamente a ellos.
   - Se implantaran varios constructores: un constructor por defecto, un constructor con el nombre, edad y sexo, un constructor con todos los atributos como parámetro (lo que se conoce como sobrecarga de constructores).
   - Los métodos que se implementarán son:
     a) calcularIMC(): Utilizar la formula $\frac{kg}{altura^2}$. Si la función devuelve un valor entre 16 y 18 debe regresar -1(infrapeso), si la formula devuelve un valor entre 18 y 25 la función debe regresar el valor de 0 (peso ideal) y si la función regresa mayor a 25, la función regresa el valor de 1 (sobrepeso).
     b) esMayorDeEdad(): indica si es mayor de edad, devuelve un booleano.
     c) comprobarSexo(char sexo): comprueba que el sexo introducido es correcto.Si no es correcto, sera H. No sera visible al exterior.
     d) toString(): devuelve toda la información del objeto.

4. Utilizando la Clase anterior, crea una clase ejecutable que haga lo siguiente:

   - Pide por teclado el nombre, la edad, sexo, peso, altura y matricula
   - Crea 2 objetos de la clase anterior, el primer objeto obtendrá las anteriores variables pedidas por teclado,el segundo objeto obtendrá todos los anteriores menos el peso y la altura, para este último utiliza los métodos para darle a los atributos un valor
   - Para cada objeto, deberá comprobar si esta en su peso ideal, tiene sobrepeso o por debajo de su peso ideal con un mensaje

- Indicar para cada objeto si es mayor de edad.
- Por último, mostrar la información de cada objeto.

Unidad 3

Relaciones entre clases

Contenido de la Unidad

3.1. Agregación y composición

La diferencia radica en el nivel de interdependencia y la independencia de los objetos en la relación. En la agregación, los objetos pueden existir independientemente, mientras que, en la composición, los objetos están íntimamente conectados y uno no puede existir sin el otro en el contexto de la relación.

1. Menciona las diferencias entre *Agregación* y *Composición*

2. Realiza una clase de nombre Compra, para guardar información sobre las compras realizadas en una tienda de conveniencia. La información que se guarda de una compra es la siguiente: clave de la transacción, la fecha y hora. Para guardar la fecha y la hora, se debe realizar dos clases, Fecha y Hora.

3.2. Herencia

1. Considere la clase Villano (como se muestra debajo). Realice una clase en lenguaje Java llamada Malvado que herede de Villano y que tenga adicionalmente un atributo llamado extraterrestre de tipo booleano (que indique si es extraterrestre o no). Agregar constructor que acepte todos sus atributos para inicializar. Adicionalmente realice una aplicación que genere 1 objetos de tipo Villano y 2 objetos de tipo Malvado en un solo arreglo. Recorrer el arreglo para imprimir el nombre y edad de cada uno.

Listing 3.1: Clase Villano.java

```
1  public class Villano {
2      private int edad;
3      private String nombre;
4
5
6  public Villano(int e, String n) {
7  edad = e;
8  nombre = n;
9  }
10
11 public int getEdad() {
12 return edad;
13 }
14
15 public String getNombre(){
16         return nombre;
17 }
18
19 }
```

2. Se plantea desarrollar un programa Java que permita la gestión de una empresa agroalimentaria que trabaja con tres tipos de productos: productos frescos, productos refrigerados y productos congelados. Todos los productos llevan esta información

común: número de lote y país de origen. A su vez, cada tipo de producto lleva alguna información específica. Los productos frescos deben llevar el año de envasado. Los productos refrigerados deben llevar el código del organismo de supervisión alimentaria. Los productos congelados deben llevar temperatura de mantenimiento recomendada.
Hay dos tipos de productos congelados: congelados por agua y congelados por nitrógeno. Los productos congelados por agua deben llevar la información de la salinidad del agua con que se realizó la congelación en gramos de sal por litro de agua. Los productos congelados por nitrógeno deben llevar la información del método de congelación empleado.
Crear el código de las clases Java implementando una relación de herencia siguiendo estas indicaciones:

- Crear superclases intermedias (aunque no se correspondan con la descripción dada de la empresa) para agrupar atributos y métodos cuando sea posible. Esto corresponde a "realizar abstracciones" en el ámbito de la programación, que pueden o no corresponderse con el mundo real.
- Cada clase debe disponer de constructor y permitir establecer (set) y recuperar (get) el valor de sus atributos y tener un método que permita mostrar la información del objeto cuando sea procedente
- Crear una clase *JJavaProductos* con el método main donde se creen: dos productos frescos, tres productos refrigerados y tres productos congelados (2 de ellos congelados por agua y uno por nitrógeno). Mostrar la información de cada producto por pantalla.

3.3. Polimorfismo

1. Identifique los dos tipos de polimorfismo (*Over-loading* y *Over-riding*) en el siguiente código.

Listing 3.2: Polimorfismo

```
1  public class Calculo
2  {
3      public int suma(int a, int b){
4          return a+b;
5      }
6
7      public int suma(int a, int b, int c){
8          return a+b+c;
9       }
10
11     public String toString(){
```

Sobre carga

Sobre escritura

```
12          return "Programa suma";
13      }
14
15  }
```

2. Implementa una clase con el nombre de Figura (clase abstracta) la cual tendrá la siguiente información: nombreFigura(string). De la cual se derivan las clases Cuadrado y Triángulo. La clase Cuadrado tiene los siguientes datos: lado(float) y la clase Triángulo:base(float), altura(float).
   La clase Figura tiene dos métodos abstractos llamados pide_Datos e imprime_Area, los cuales realizan lo siguiente: el primero pide los datos para calcular el área de acuerdo al objeto que lo invoca y el segundo calcula e imprime el área del objeto que lo invoca. Así mismo, implementa dentro de la clase principal (Polimo) un método llamado "Imprime" que recibirá como argumento un objeto de la superclase Figura, y dentro de la cual, el objeto pasado como argumento invocará a los métodos imprime_Area, recuerda que esto es un claro ejemplo de polimorfismo. De igual forma sobrecarga el método toString en las clases realizadas, así como los constructores de las clases derivadas, las cuales podrán recibir los datos para calcular el área acorde a la figura. En la clase principal, declara un arreglo de variables de la clase base y genera código para invocar a los métodos abstractos y comprobar que funcione de forma correcta el método imprime

3.4. Interfaces

1. Subraya la respuesta correcta de las opciones presentadas, algunas preguntas puede ser de tema vistos con anterioridad.

   - La palabra *extends* aplica cuando.
     a) Una clase hereda de otra clase
     b) Una variable
     c) Un método
     d) Una expresión
   - Una clase Java puede heredar de dos o más clases.
     a) Verdadero.
     b) Falso
   - En Java, una herencia múltiple es implementada usando el concepto de :
     a) Una interface
     b) Una clase abstracta
     c) Una clase privada

- ¿Cuál de las siguientes atributos son heredados por una subclase?
  - *a*) Todas los atributos y métodos de la clase padre
  - *b*) Solo los atributos y métodos privados de la clase padre
  - *c*) Todos los atributos y métodos protegidos de la clase padre
- ¿Qué palabra reservada es usada en el encabezado de la clase cuando una clase es definida como subclase de una interfaz?
  - *a*) inherits
  - *b*) include
  - *c*) extends
  - *d*) implements
- ¿Cómo puedes instanciar un objeto de una clase abstracta?
  - *a*) Con un constructor
  - *b*) Con un constructor por defecto
  - *c*) No es posible instanciar un objeto de una clase abstracta
- ¿Puede una clase de Java implementar una o más interfaces?
  - *a*) Falso
  - *b*) Verdadero
- Cuando una clase realiza polimorfismo sobre un método del tipo overrides, ¿qué objeto de referencia es usado para llamar al método heredado de la superclase?
  - *a*) class
  - *b*) super
  - *c*) inherited
- Si una clase contiene un método abstracto, entonces
  - *a*) la clase debe ser declarada como abstracta
  - *b*) la clase no es abstracta
  - *c*) la clase podría o no podría ser abstracta
- Que puedes decir del siguiente método 'public void myMetodo(){}'
  - *a*) Este método no es abstracto
  - *b*) Este método es abstracto

2. Realice una clase que herede de la clase Villano (mostrado en el ejercicio que se encuentra en la sección de Herencia) llamada VillanoDeUltratumba que tenga un atributo adicional de tipo String llamado legión (que se ponga valor inicial en su constructor) y que implemente una interfaz llamada SerDeUltratumba la cual requiera 2 métodos: asustar (que debe imprimir "buuuuu!!!!") y gritar (que debe imprimir "uaaaaay!!"). Realice las clases o interfaces que se necesiten.

3.5. Diagramas UML

1. A partir de la Figura 3.1 realiza el código en Java

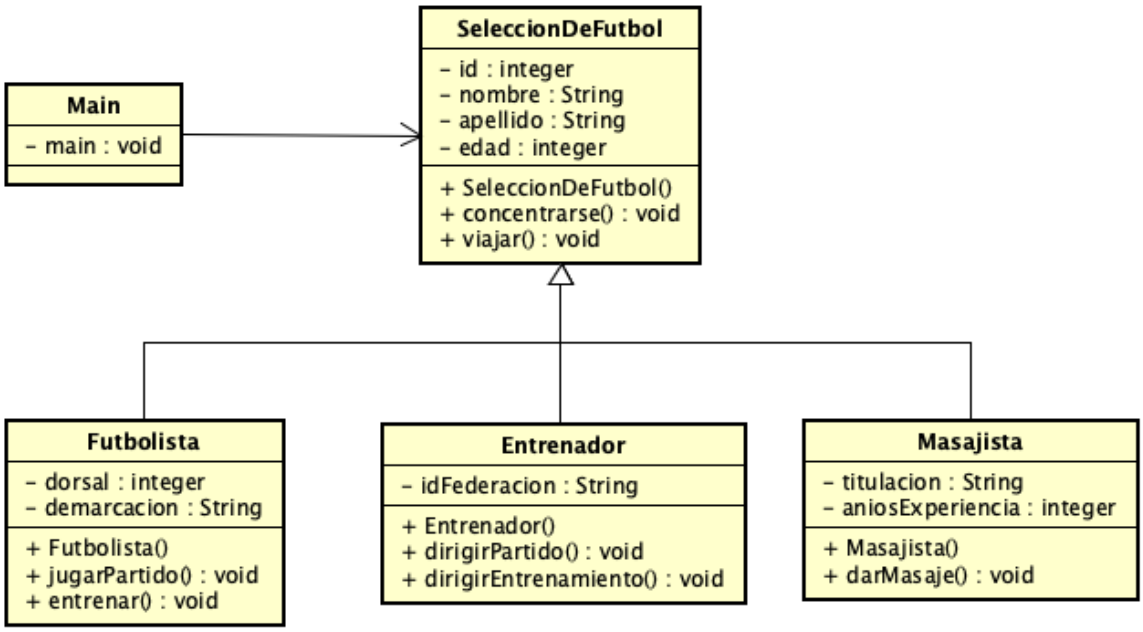

Figura 3.1: Realiza el codigo en Java a partir del Diagrama de Clases

2. A partir de la Figura 3.2 realiza el código en Java

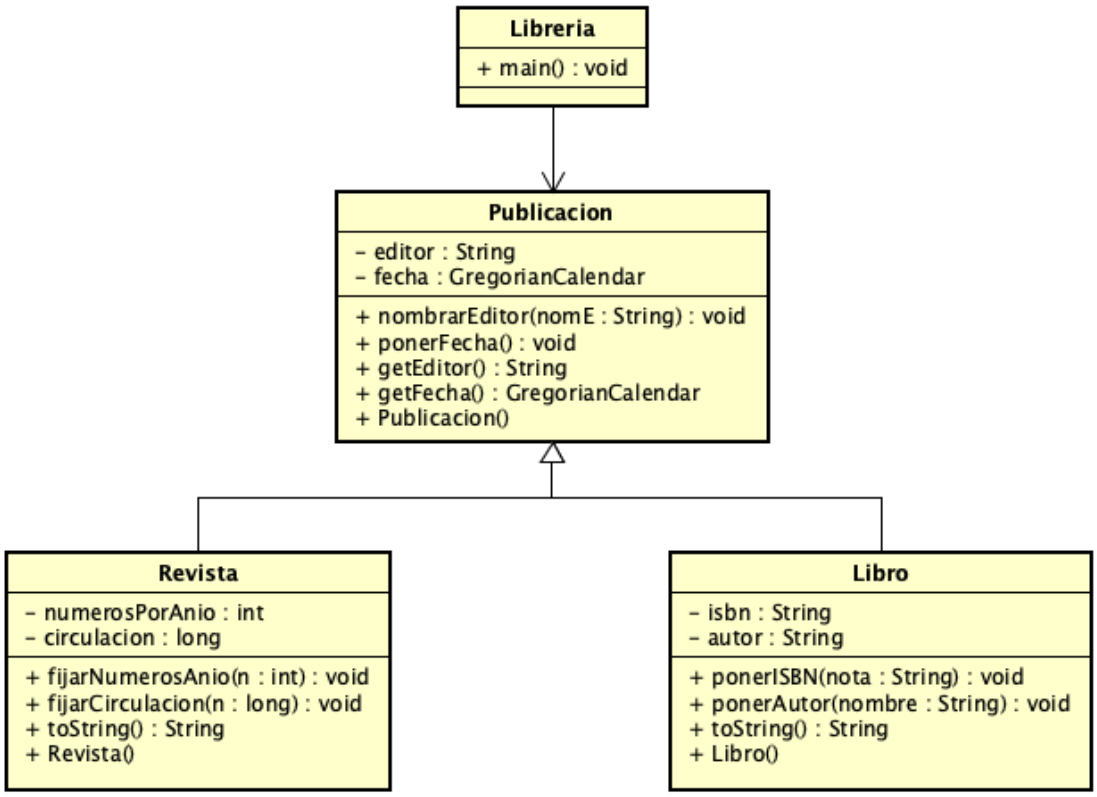

Figura 3.2: Realiza el código en Java a partir del Diagrama de Clases

3. Realiza el diagrama de clases del ejercicio 2 correspondiente a la sección de Herencia.

4. Realiza el diagrama de clases del ejercicio 1 correspondiente a la sección de Herencia y el ejercicio 2 correspondiente a la sección de Interfaces.

Unidad 4

Manejo de errores y excepciones

Contenido de la Unidad

4.1. Excepciones y errores

1. Ejercicios de opción múltiple.

   - ¿Por qué son útiles los bloques try / catch ?
     a) Pueden reemplazar las declaraciones de selección, ahorrando así tiempo de CPU.
     b) Permite a los programadores recuperarse de situaciones ilegales y continuar ejecutando el programa.
   - Algunos métodos que lanzan una excepción requieren bloques try y catch , mientras que otros no.
     a) Verdadero
     b) Falso
   - ¿Qué palabra clave se encuentra en el encabezado de un método que podría detectar un error y generar una excepción apropiada?
     a) throw
     b) throws
     c) exception
     d) exceptions
   - Al codificar un bloque try and catch , es obligatorio codificar un bloque *finally*
     a) Verdadero
     b) Falso
   - La mayoría de las clases relacionadas con entradas y salidas se pueden encontrar en el paquete
     a) java.file
     b) java.inputoutput
     c) java.io
     d) java.readwrite
   - Si abrimos un archivo para leer y el archivo no existe,
     a) Error de compilación
     b) Se lanza una excepción
     c) El archivo se crea automáticamente
   - Si el código en el bloque try puede generar múltiples tipos de excepciones, podemos proporcionar múltiples bloques catch , uno para cada posible excepción,
     a) Falso
     b) Verdadero

2. Basandose en el problema 2 valida con un *try* y *catch* que el usuario no introduzca para el caso de la clase Triángulo que la altura sea mayor que la base y para el caso del Cuadrado que el lado sea mayor a 5. Para hacer lo anterior, es necesario que generes una clase de excepción llamada *ExceptionFigura* y utilices *throw* y *throws* convenientemente . El mensaje de advertencia al usuario es *La base debe ser mayor o igual a la altura* para el caso del Triángulo, y para caso del Cuadrado *El lado debe ser mayor a 5*.

4.2. Tipo de excepciones

1. Relaciona los tipos de excepciones con su descripción

   ( ) Lanza una excepción cuando el fichero no se encuentra

   ( ) Lanza una excepción cuando no existe la clase.

   ( ) Lanza una excepción cuando llega al final del fichero..

   ( ) Lanza una excepción cuando se accede a una posición de un array que no exista.

   ( ) Lanza una excepción cuando se procesa un numero pero este es un dato alfanumérico.

   ( ) Lanza una excepción cuando intentando acceder a un miembro de un objeto para el que todavía no hemos reservado memoria.

   ( ) Generaliza muchas excepciones anteriores. La ventaja es que no necesitamos controlar cada una de las excepciones.

   ( ) Es la clase padre de IOException y de otras clases. Tiene la misma ventaja que IOException.

   1 -ClassNotFoundException

   2 -ArrayIndexOutOfBoundsException

   3 -IOException

   4 -NullPointerException

   5 -Excepcion

   6 -FileNotFoundException

   7 -NumberFormatException

   8 -EOFException

2. Realiza un programa en Java para leer un archivo *datos.txt*, utiliza los bloques *try* y *catch* y el tipo excepción adecuada para capturar la excepción de archivo no encontrado.

4.3. Aserciones

Responde correctamente :

1. Define que es una aserción.
2. ¿Cuál es el objetivo de programar utilizando aserciones?
3. ¿Cuál es la palabra reservada para crear aserciones?
4. ¿Cuál es la parámetro que se utiliza al momento de compilar para que las aserciones puedan funcionar?
5. Escribe un ejemplo de una aserción donde se requiera controlar que el valor de una variable esté dentro de un rango determinado.

Unidad 5

Elementos de diseño para la POO

Contenido de la Unidad

5.1. Persistencia de datos

1. Desarrolla un programa que escriba en un archivo objetos de tipo Cliente, donde el Cliente tiene los siguientes atributos: Clave (int), Nombre(String), Apellido(String),LineadeCredito(double).

2. Desarrolla un programa que lea de un archivo objetos de tipo Cliente, definidos en el ejercicio anterior.

5.2. Interfaces gráficas de usuario (IDE)

1. Subraya la respuesta correcta.

   - Un ejemplo de una clase de componente GUI es
     a) FXML.
     b) TextField
     c) Controlador
     d) Stage
   - ¿En qué paquete encuentra las clases Button, TextField y ComboBox ?
     a) javafx.scene
     b) javafx.scene.control
     c) java.scene
     d) java.awt
   - La propiedad de un elemento Label que especifica el texto dentro de la etiqueta es
     a) Label.
     b) text
     c) Word
     d) Phrase
   - La propiedad que especifica el nombre de la clase Controlador para la Vista definida en el documento FXML es
     a) control
     b) fx:control
     c) controlador
     d) fx:controlador
   - La propiedad de un elemento Button que especifica el método llamado cuando el usuario hace clic en el botón es
     a) Action

b) Press
c) onAction
d) onPress

- ¿Qué atributo y anotación utilizamos con una variable de instancia del Controlador para hacer referencia a un componente GUI definido en un documento FXML?

  a) id y FXML.
  b) id y @FXML
  c) fx:id y FXML
  d) fx:id y @FXML

- Suponga que hemos definido correctamente el atributo FXML de un elemento Button que especifica el método a llamar cuando el usuario hace clic en ese botón. ¿Cuál es el tipo de retorno de ese método?

  a) Button
  b) boolean
  c) onAction
  d) void

2. Realiza un programa utilizando JavaFX y la arquitectura MVC que le permita un usuario calcular el cubo y el cuadrado de un número.

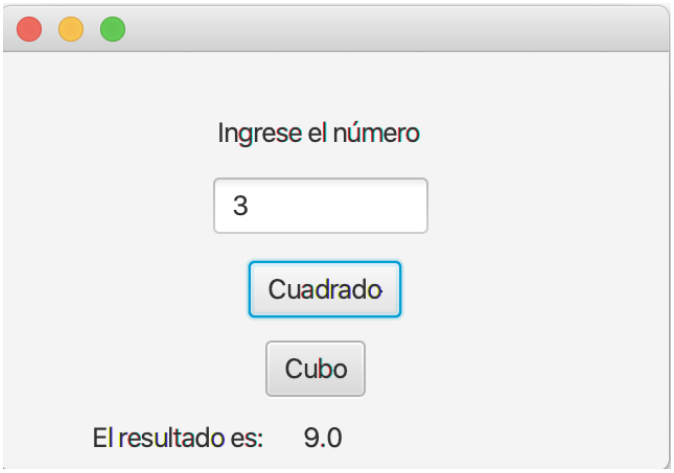

Figura 5.1: Vista del programa que calcula el cuadro y el cubo de un número

3. Realiza un programa utilizando JavaFX y la arquitectura MVC que le permita a un usuario que a partir de un archivo de una imagen pueda variar el nivel de umbralización para su binarización. La binarización consiste en que los pixeles solo pueden tomar dos valores (blanco / negro) y la umbralización se refiere que dado

un valor de umbral u los valores de los pixeles de la imagen original que sea menor a ese valor cambian a negros y los mayores a u se convierten en blancos. Utiliza un control *slider* para modificar los valores de u.

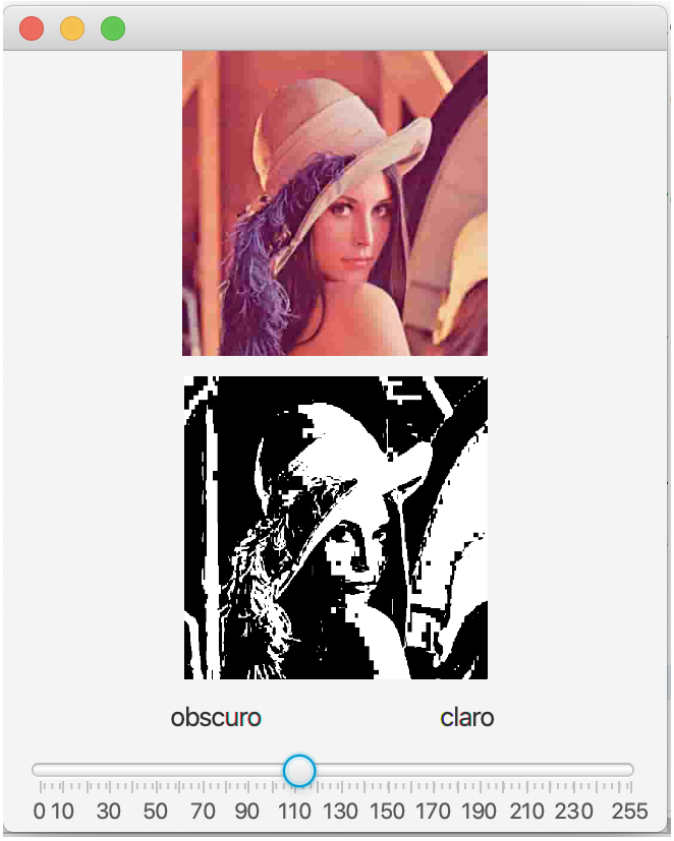

Figura 5.2: Vista del programa que binariza una imagen con respecto a un umbral de 0 a 255

5.3. Arquitectura por Capas

1. Relaciona los principios de SOLID con su descripción

   ( ) Una entidad de software debería estar abierta a extensión pero cerrada a modificación

   ( ) Si en alguna parte de nuestro código estamos usando una clase, y esta clase es extendida, tenemos que poder utilizar cualquiera de las clases hijas y que el programa siga siendo válido.

   ( ) Un objeto debe realizar una única cosa

   ( ) Podemos hacer que el código que es el núcleo de nuestra aplicación no dependa de los detalles de implementación, como pueden ser el framework que utilices, la base de datos,etc.

( ) Ninguna clase debería depender de métodos que no usa.

1 -Principio de Responsabilidad única

2 -Principio Open-Closed

3 -Principio de Sustitución de Liskov

4 -Principio de Segregación de Interfaces

5 -Principio de Inversión de Dependencias

2. Describe tres situaciones que nos pueden ayudar a detectar que el Principio de Responsabilidad Única no se esta cumpliendo?

3. ¿Cómo podemos detectar si estamos faltando al Principio de sustitución de Liskov?

4. Explica el principio de Open/Closed.

5. Proporciona un diagrama que ilustre de una forma sencilla el MVC

6. Relaciona los componentes del MVC con su descripción

Componente	Descripción
( )Controlador	1-Presenta la información al usuario (*frontend*)
( )Vista	2-Es la representación de los datos
( )Modelo	3-Responde a eventos e invoca peticiones

Bibliografía

[1] Weaver J, Vos J, Chin S (2019) *The Definitive Guide to Modern Java Clients with JavaFX: Cross-Platform Mobile and Cloud Development.*Apress.

[2] Taman M, (2015)*JavaFX Essentials.*Packt Publishing.

[3] Schildt H (2015)*Introducing JavaFX 8 Programming.*McGraw-Hill.

[4] Anderson, (2018) *Java Illuminated, 5t Edition.* Jones & Bartlett Learning.

[5] Martín, R.C., (2003) *UML for Java programmers.*Prentice Hall.

[6] Deitel, H, y Deitel P (2004) *Java como programar 5 ed..*Prentice Hall.