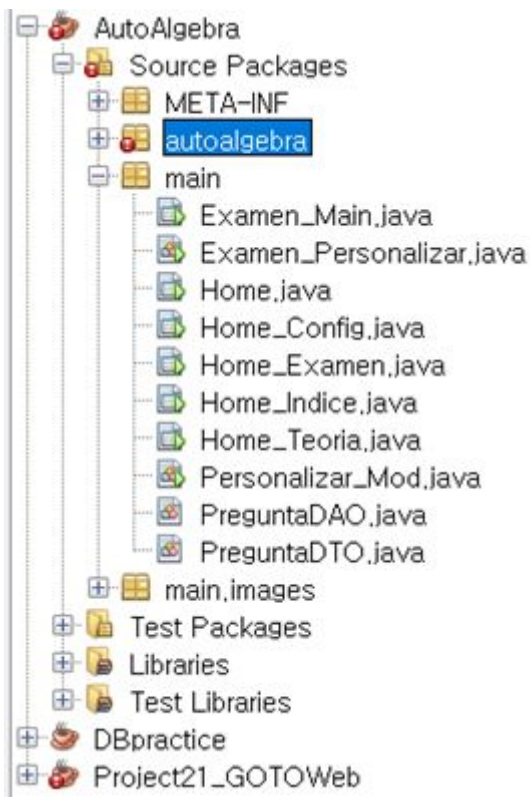


Criterio C: Desarrollo**Proyecto: AutoAlgebra****Entorno de desarrollo integrado:
NetBeans IDE 8.2****Todas las clases:**

El uso del constructor

Un constructor es un método especial de una clase o estructura en la programación orientada a objetos que inicializa un objeto de ese tipo. Un constructor es un método de instancia que generalmente tiene el mismo nombre que la clase y se puede usar para asignar los valores de los miembros de un objeto, ya sea por defecto o por valores definidos por el usuario. En el programa se implementó en la clase `Personalizar_Mod`.



```

28
29 public Personalizar_Mod(){//constructor para agregar pregunta
30
31     createUI(); // Método para crear IU
32     btnActualizar.setEnabled(false);
33     btnActualizar.setVisible(false);
34     btnBorrar.setEnabled(false);
35     btnBorrar.setVisible(false);
36
37 }
38 //constructor
39
40 public Personalizar_Mod(Examen_Personalizar pList){ //constructor para agregar pregunta
41
42     createUI();// Método para crear IU
43     btnActualizar.setEnabled(false);
44     btnActualizar.setVisible(false);
45     btnBorrar.setEnabled(false);
46     btnBorrar.setVisible(false);
47     this.pList = pList;
48

```

En la figura se observa constructores básicos para abrir el interfase de usuario para agregar una pregunta. Está compuesto por el método “invocado” de creador UI (Método que crea la interfase de usuario), botones para actualizar y borrar, y `pList` como auto referencia como objeto.

Manejo de datos y manejo de excepciones

Es común tener errores lógicos al ejecutar los códigos. Para evitar, el lenguaje JAVA obliga al programador manejar excepciones en ciertas ocasiones. Uno de ellas es el manejo de datos a través de BD. Para manejar el error, se implementó *try-catch*.

```

49  /**Método para recibir datos de una pregunta*/
50  public PreguntadTO getMemberDT0(int id_pregunta){
51
52      PreguntadTO dto = new PreguntadTO();
53
54      Connection con = null;      //conexión
55      PreparedStatement ps = null; //dar orden
56      ResultSet rs = null;       //resultado
57
58      try {
59
60          con = getConn();
61          String sql = "select * from pregunta where id_pregunta=?";
62          ps = con.prepareStatement(sql);
63          ps.setInt(1, id_pregunta);
64
65          rs = ps.executeQuery();
66
67          if(rs.next()){
68              dto.setId(rs.getInt("id_pregunta"));
69              dto.setPr(rs.getString("pregunta"));
70              dto.setRe(rs.getString("respuesta"));
71              dto.setPt(rs.getString("puntaje"));
72              dto.setRf1(rs.getString("r_falsa1"));
73              dto.setRf2(rs.getString("r_falsa2"));
74              dto.setRf3(rs.getString("r_falsa3"));
75          }
76      } catch (Exception e) {
77          e.printStackTrace();
78      }
79
80      return dto;
81  }
82
83

```

Se puede observar la introducción de SQL, que permite acceder a la BD y ordenar. Otro aspecto de complejidad es el uso de *try-catch*. Se utilizó un método común de `printStackTrace()` para manejar errores que podrían ser producidos durante la ejecución del programa. Este método se utiliza para imprimir el registro del stack donde se ha iniciado la excepción. Gracias a este método se logró aislar el código para manejar errores del código de manejo de datos, el cual evitará el fallo total del sistema.

Manejo de Loop anidado

```

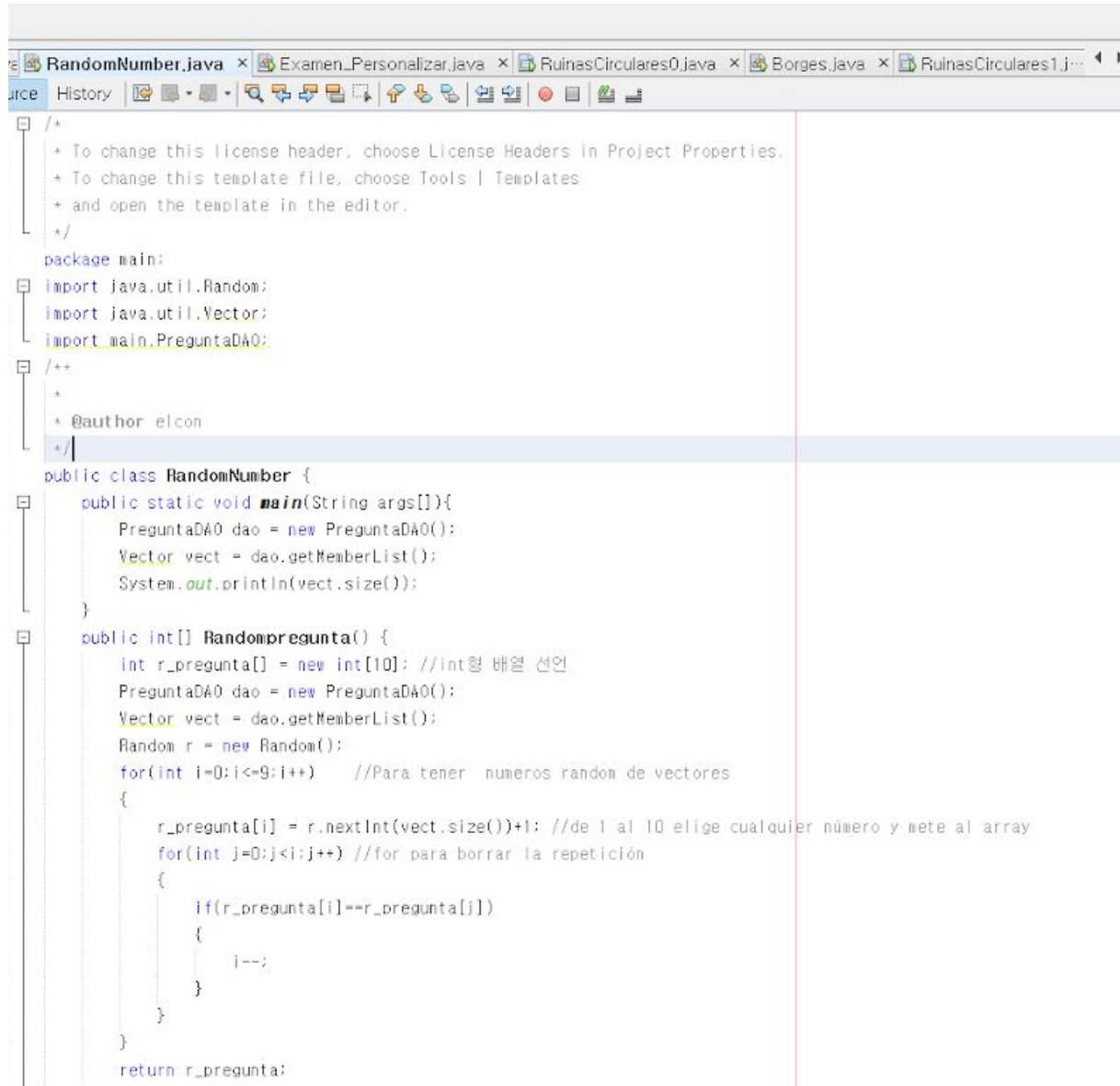
1  import main.PreguntadAO;
2
3  /**
4   * @author elcon
5   */
6  public class RandomNumber {
7      public static void main(String args[]){
8          PreguntadAO dao = new PreguntadAO();
9          Vector vect = dao.getMemberList();
10         System.out.println(vect.size());
11     }
12
13     public int[] Randompregunta() {
14         int r_pregunta[] = new int[10]; //int형 배열 선언
15         PreguntadAO dao = new PreguntadAO();
16         Vector vect = dao.getMemberList();
17         Random r = new Random();
18         for(int i=0;i<=9;i++) //Para tener numeros random de vectores
19         {
20             r_pregunta[i] = r.nextInt(vect.size())+1; //de 1 al 10 elige cualquier número y mete al array
21             for(int j=0;j<i;j++) //for para borrar la repetición
22             {
23                 if(r_pregunta[i]==r_pregunta[j])
24                 {
25                     j--;
26                 }
27             }
28         }
29         return r_pregunta;
30     }
31 }

```

Se puede observar un *loop* dentro del *loop* con el fin de poner números aleatorios que no repitan. El *loop* grande que cubre el *loop* pequeño, hasta que no acabe sus vueltas no

puede proceder el *loop*. El *loop* pequeño tiene función de comparar si el *array* anterior es igual al *array* que agregó. Si es igual, se reduce la variable *i* y se repite hasta la introducción del valor diferente a los *arrays* pasados. Esto permite cada vez al realizar el examen tenga un orden distinto con respuestas de diferentes posiciones.

Manejo de la librería



```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package main;
import java.util.Random;
import java.util.Vector;
import main.PreguntaDAO;

/**
 *
 * @author elcon
 */
public class RandomNumber {

    public static void main(String args[]){
        PreguntaDAO dao = new PreguntaDAO();
        Vector vect = dao.getMemberList();
        System.out.println(vect.size());
    }

    public int[] Randompregunta() {
        int r_pregunta[] = new int[10]; //int형 배열 선언
        PreguntaDAO dao = new PreguntaDAO();
        Vector vect = dao.getMemberList();
        Random r = new Random();
        for(int i=0;i<=9;i++) //Para tener numeros random de vectores
        {
            r_pregunta[i] = r.nextInt(vect.size())+1; //de 1 al 10 elige cualquier número y mete al array
            for(int j=0;j<i;j++) //for para borrar la repetición
            {
                if(r_pregunta[i]==r_pregunta[j])
                {
                    i--;
                }
            }
        }
        return r_pregunta;
    }
}

```

se maneja la librería de vector (*array*) y random (método para tener orden aleatorio a ciertas variables como integral). Es necesario para el programa, porque la librería vector se encarga de transportar y almacenar todos los datos y random se encarga de tener un orden aleatorio.

Manejo de loop

En la mayoría de los lenguajes de programación informática, un *loop while* es un tipo de bucle que ayuda a que el código se repita en función de una condición booleana dada. Este bucle *while* puede considerarse como una iteración de una sentencia *if*.

```

Connection con = null; //연결
PreparedStatement ps = null; //명령
ResultSet rs = null; //결과

try{
    con = getConn();
    //String sql = "select * from pregunta order by name asc";
    String sql = "select * from pregunta";
    ps = con.prepareStatement(sql);
    rs = ps.executeQuery();

    while(rs.next()){
        int id_pregunta = rs.getInt("id_pregunta");
        String pregunta = rs.getString("pregunta");
        String respuesta = rs.getString("respuesta");
        String puntaje = rs.getString("puntaje");
        String r_falsa1 = rs.getString("r_falsa1");
        String r_falsa2 = rs.getString("r_falsa2");
        String r_falsa3 = rs.getString("r_falsa3");

        Vector row = new Vector();
        row.add(id_pregunta);
        row.add(pregunta);
        row.add(respuesta);
        row.add(puntaje);
        row.add(r_falsa1);
        row.add(r_falsa2);
        row.add(r_falsa3);

        data.addRow();
    }
} catch (Exception e) {
    e.printStackTrace();
}

```

Se puede observar un *while* para insertar al vector todos los datos a la posición del *array* asignado. Se utiliza para importar los datos desde esta clase hasta otras. Esta complejidad ocupa un rol muy importante, ya que es el dato para modificar y se puede modificar dentro de las bases de datos.

Palabras contadas: 518

Referencias:

Techopedia, S. (2019). What is a Constructor? - Definition from Techopedia. Recuperado en Febrero 1, 2019, de <https://www.techopedia.com/definition/5656/constructor>