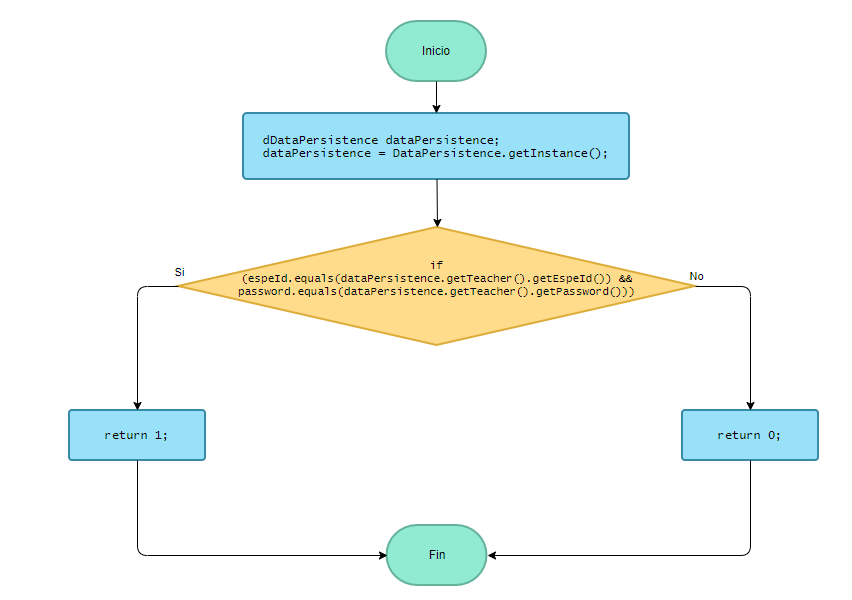
|  |
| --- |
|  |
| Prueba de Caja Blanca |
| ***“Gestión de información académica InClass”*** |
|  |
|  |
| **Integrantes:**  Stephen Drouet  Bryan Morales  Alejandro Sarmiento  Jairo Quilumbaquin  **Fecha 28/02/2024** |

**Prueba caja blanca login**

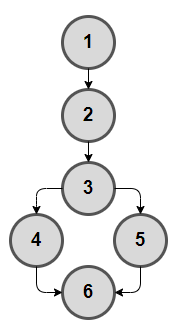
**CÓDIGO FUENTE**

|  |
| --- |
| public static int loginTeacher(String espeId, String password) {  DataPersistence dataPersistence;  dataPersistence = DataPersistence.getInstance();  if (espeId.equals(dataPersistence.getTeacher().getEspeId()) && password.equals(dataPersistence.getTeacher().getPassword())) {  return 1;  } else {  return 0;  }  } |

**DIAGRAMA DE FLUJO**

****

**Grafo**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 6

**R2:** 1, 2, 3, 5, 6

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=1+1=2

* V(G) = A – N + 2

V(G)= 6 – 6 + 2 = 2

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos

**Prueba caja blanca navegar menu**

**CÓDIGO FUENTE**

|  |
| --- |
| private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt)  private void menuEnrolledActionPerformed(java.awt.event.ActionEvent evt)  private void menuAddStudentActionPerformed(java.awt.event.ActionEvent evt)  private void menuRemoveStudentActionPerformed(java.awt.event.ActionEvent evt)  private void menuManageGradeActionPerformed(java.awt.event.ActionEvent evt)  private void menuGradeRecordActionPerformed(java.awt.event.ActionEvent evt)  private void menuTakeAttendanceActionPerformed(java.awt.event.ActionEvent evt)  private void menuGetAttendaceActionPerformed(java.awt.event.ActionEvent evt)  private void menuTutorshipsActionPerformed(java.awt.event.ActionEvent evt)  private void menuManageCoursesActionPerformed(java.awt.event.ActionEvent evt)  if (evt) {  FrmX frmX = new FrmX();  frmX.setVisible(true);  this.setVisible(false);  } |

**DIAGRAMA DE FLUJO**

**Diagrama

Descripción generada automáticamente**

**GRAFO**

**Forma, Flecha, Círculo

Descripción generada automáticamente**

**RUTAS**

**R1:** 1, 2, 3, 4, 5

**R2:** 1, 2, 3, 2, 3, 4, 5

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=2+1=3

* V(G) = A – N + 2

V(G)= 5– 5 + 2 = 2

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos

**Prueba caja blanca calcular porcentaje de asistencias**

**CÓDIGO FUENTE**

|  |
| --- |
| public static float calculateAssistancePersetn(Student student) {  float assistancePersent;  int assistances = 0;  student.getAttendanceRecord().setTotalClassNumber(student. getAttendanceRecord().getAttendance().size());  for (Boolean dayAssistance : student.getAttendanceRecord(). getAttendance()) {  if (dayAssistance) {  assistances++;  }  }  assistancePersent = (float) assistances / (float) student.getAttendanceRecord().getTotalClassNumber();  assistancePersent = Math.round(assistancePersent \* 10000) / 100;  return assistancePersent;  } |

**DIAGRAMA DE FLUJO**

**Diagrama

Descripción generada automáticamente**

**GRAFO**

**Un dibujo de un reloj

Descripción generada automáticamente con confianza media**

**RUTAS**

**R1:** 1, 2, 3, 4, 7, 8

**R2:** 1, 2, 3, 4, 5, 4, 7, 8

**R3:** 1, 2, 3, 4, 5, 6, 4, 7, 8

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=2+1=3

* V(G) = A – N + 2

V(G)= 9– 8 + 2 = 3

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

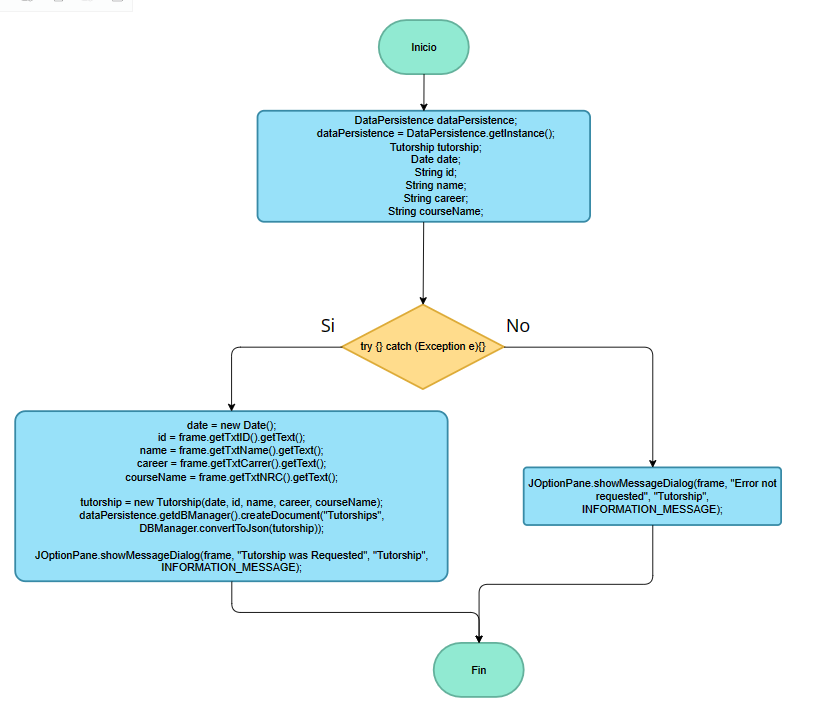
**N:** Número de nodos

**Prueba caja blanca solicitar tutorías**

**CÓDIGO FUENTE**

|  |
| --- |
| DataPersistence dataPersistence;  dataPersistence = DataPersistence.getInstance();  Tutorship tutorship;  Date date;  String id;  String name;  String career;  String courseName;  try {  date = new Date();  id = frame.getTxtID().getText();  name = frame.getTxtName().getText();  career = frame.getTxtCarrer().getText();  courseName = frame.getTxtNRC().getText();  tutorship = new Tutorship(date, id, name, career, courseName);  dataPersistence.getdBManager().createDocument("Tutorships", DBManager.convertToJson(tutorship));  JOptionPane.showMessageDialog(frame, "Tutorship was Requested", "Tutorship", INFORMATION\_MESSAGE);  } catch (Exception e) {  JOptionPane.showMessageDialog(frame, "Error not requested", "Tutorship", INFORMATION\_MESSAGE);  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

**Forma, Círculo

Descripción generada automáticamente**

**RUTAS**

**R1:** 1, 2, 3, 4, 6

**R2:** 1, 2, 3, 5, 6

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=1+1=2

* V(G) = A – N + 2

V(G)= 6– 6 + 2 = 2

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

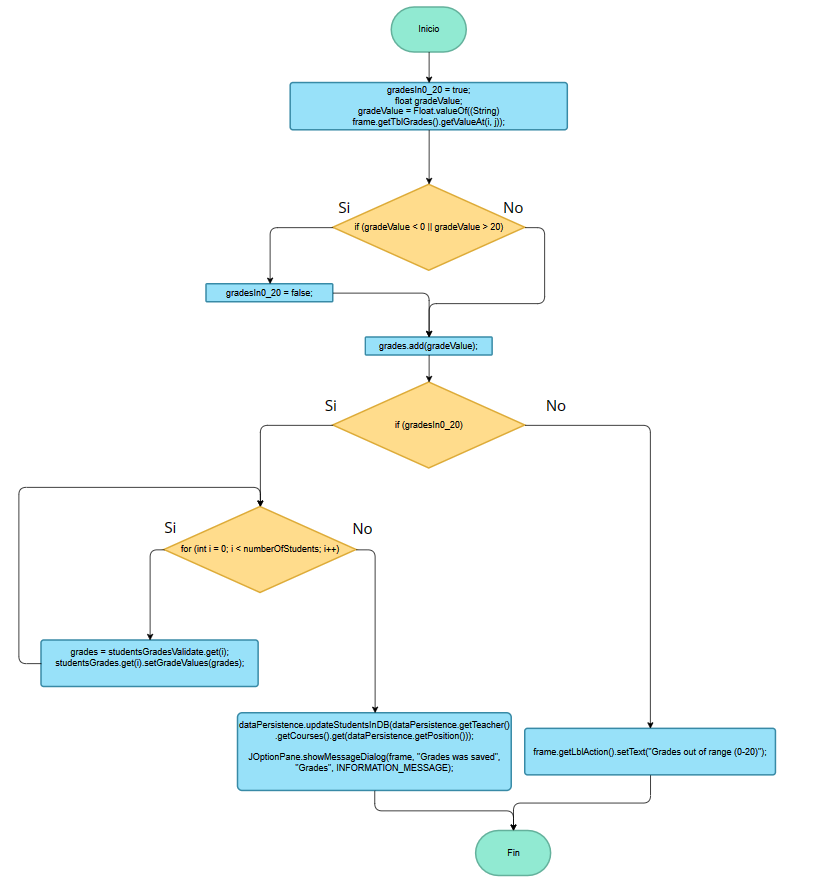
**N:** Número de nodos

**Prueba caja blanca ingresar notas**

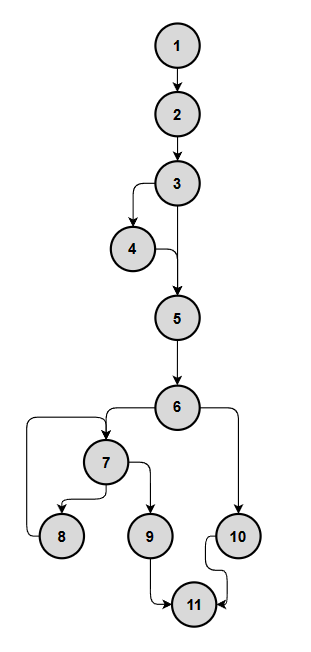
**CÓDIGO FUENTE**

|  |
| --- |
| gradesIn0\_20 = true;  float gradeValue;  gradeValue = Float.valueOf((String) frame.getTblGrades().getValueAt(i, j));  if (gradeValue < 0 || gradeValue > 20) {  gradesIn0\_20 = false;  }  grades.add(gradeValue);  if (gradesIn0\_20) {  for (int i = 0; i < numberOfStudents; i++) {  grades = studentsGradesValidate.get(i);  studentsGrades.get(i).setGradeValues(grades);  }  dataPersistence.updateStudentsInDB(dataPersistence.getTeacher().getCourses().get(dataPersistence.getPosition()));  JOptionPane.showMessageDialog(frame, "Grades was saved", "Grades", INFORMATION\_MESSAGE);  frame.getLblAction().setText("");  } else {  frame.getLblAction().setText("Grades out of range (0-20)");  } |

**DIAGRAMA DE FLUJO**

****

**GRAFO**

****

**RUTAS**

**R1:** 1, 2, 3, 4, 5, 6, 10, 11

**R2:** 1, 2, 3, 5, 6, 7, 9, 11

**R3:** 1, 2, 3, 5, 6, 7, 8, 7, 9, 11

**R4:** 1, 2, 3, 4, 5, 10, 11

**COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

* V(G) = número de nodos predicados(decisiones)+1

V(G)=3+1=4

* V(G) = A – N + 2

V(G)= 13– 11 + 2 = 4

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos