PROGRAMACIÓN AVANZADA I

2do. Parcial

Junio - 2024

1-Responder Verdadero o Falso según corresponda, Justificar en caso de Falso

a) Al definir un constructor de clase sin modificador de acceso, es decir package-private, significa que solamente se lo puede invocar desde ta misma clase Desde el paquete

b) El método ***split()*** de la clase String elimina los espacios en blanco en ambos lados de un string Corta y hace array

c) Las siglas del patrón DAO significan Data Abstract Object

d) Singleton es un patrón de diseño que garantiza que tan solo exista un objeto de su tipo y proporciona un único punto de acceso a él para cualquier otro código.

2- Completa dando los valores a los "?#" según correspondan:

```
public class DAOAlumnoFactory {
   public static final String TIPO_DAO = "TIPO_DAO";
   public static final String DAO_TXT = "DAO_TXT";
   public static final String DAO_SQL = "DAO_SQL";
   public static final String FULL_PATH = "FULL_PATH";
   public static final String SQL_CONNECTION = "SQL_CONNECTION";
   public static final String URL_DB = "URL_DB";
   public static final String USER_DB = "USER_DB";
   public static final String PWD_DB = "PWD_DB";
public static DAO<Alumno> crearDAO(Map<String, String> ?1) throws ?2 {   DAOFactoryException
     try {                                                                configMap
       String tipoDAO = configMap.get(TIPO_DAO);
       switch (tipoDAO) {
         case DAO_TXT:
           String fullpath = configMap.get(FULL_PATH);
           return new DAOAlumnoTxt(fullpath);
         case ?3: DAO_SQL
           String url = configMap.get(URL_DB);
           String user = configMap.get(USER_DB);
           String password = configMap.get(PWD_DB);
           return new DAOAlumnoSQL(url, user, password);
         default:   DAOFactoryException
           throw new ?4("No implementado");
       }
     } catch (DAOException ?5) { ex
       Logger.getLogger(DAOAlumnoFactory.class.getName()).log(Level.SEVERE, null, ex);
       throw new DaoFactoryException("Error al crear el DAO ("
                                                  + ex.getLocalizedMessage() + ")");
     }
   }
}
```

3- Completar el siguiente código para que compile sin errores, y para que en tiempo de ejecución no arroje ninguna excepción
(No es necesario definir la variable readP5)

```
@Override
public Alumno read(Integer dni) throws DaoException {
  try {
    readPS.setInt(0, dni); 1
    ResultSet rs = readPS.executeQuery();
    if (rs.next()) {
      Alumno alumno = new Alumno();
      alumno.setDni(rs.getString ("DNI"));    getInt("DNI")
      alumno.setNombre(rs.getString("NOMBRE"));
      alumno.setFechaNac(AlumnoUtils.sqlDate2LocalDate(rs.getDate("FEC_NAC")));
      return alumno;
    }
  } catch (SQLException ex) {
    Logger.getLogger(DaoSQL.class.getName()).log(Level.SEVERE, null, ex);
    throw new DaoException("Error SQL => No se pudo leer el alumno ("
                                                    + ex.getLocalizedMessage() + ")", ex);
  } catch (PersonaException ex) {
    Logger.getLogger(DaoSQL.class.getName()).log(Level.SEVERE, null, ex);
    throw new DaoException("Error al crear el alumno ("
                                                    + ex.getLocalizedMessage() + ")", ex);
  }
return; return null;
}
```

4 - Dado el siguiente código, indicar cuales de las afirmaciones son correctas

```
@Override
public void update(Alumno alu) throws DaoException {
  try {
    long filePointer = 0;
    raf.seek(0);
    String lineaAlu;
    Integer dniAlu;
    while ((lineaAlu = raf.readLine()) != null) {
      dniAlu = Integer.valueOf(lineaAlu.substring(0, 8));
      if (dniAlu.equals(alu.getDni())) {
        raf.seek(filePointer);
        raf.writeBytes(alu.toString());
        return;
      }
      filePointer = raf.getFilePointer();
    }
  } catch (IOException ex) {
    Logger.getLogger(DaoTXT.class.getName()).log(Level.SEVERE, null, ex);
    throw new DaoException("Error E/S ==> No se pudo leer el archivo"+
      "("+ ex.getLocalizedMessage() + ")");
  }
}
```

a) La variable filePointer guarda la posición de fin de la linea con el objetivo que *raf* pueda continuar leyendo. Posicion inicial
b) La línea de código raf.seek(0); se utiliza para abrir el archivo de texto para comenzar a utilizarlo. Posiciona el puntero
c) El método getFilePointer() de RandomAccessFile me permite posicionar el puntero del archivo en la posición deseada. Trae valor puntero
d) Todas son correctas.
e) Ninguna es correcta

5-Desarrollar una clase que cumpla con el patrón de diseño Singleton:

```
public class MiSingleton {
  private static MiSingleton instancia;
  private MiSingleton() { //Inicializaciones internas que hagan falta }

  public static MiSingleton getInstancia() {
    if (instancia == null) {
      instancia = new MiSingleton();
    }
    return instancia;
  }
}
```