EXAMEN PRÁCTICO

ENTORNOS DE DESARROLLO

2ª EVALUACIÓN

CURSO: 2023/2024

RAÚL PALAO LOZANO

# ENUNCIADO PRIMER EJERCICIO (4 puntos)

Tenemos el siguiente código JAVA que le da la vuelta a un String dado:

Crea el fichero **.java** con el siguiente programa:

*public class StringUtils {*

*public static String reverse(String str) {*

*if (str == null) {*

*return null;*

*}*

*StringBuilder reversed = new StringBuilder(str).reverse();*

*return reversed.toString();*

*}*

*}*

1. Realiza una clase **StringUtilsNullTest** y otra **StringUtilsNotNullTest**.
2. En la primera deberás probar la función **usando nulo.** (1.5 puntos).
3. En la segunda deberás **realizar cinco casos de prueba**, incluyendo en uno espacios, en otro mayúsculas y en otro caracteres especiales (ñ, ç, etc.). (2 puntos).
4. Crea una **suite de test** con los dos ficheros anteriores. (0.5 puntos).

COMPRIME EN UN .ZIP LA CARPETA DEL PROYECTO Y PEGA LAS SIGUIENTES CAPTURAS:

PEGA AQUÍ CAPTURA CON EL CÓDIGO DEL APARTADO I

A screen shot of a computer program

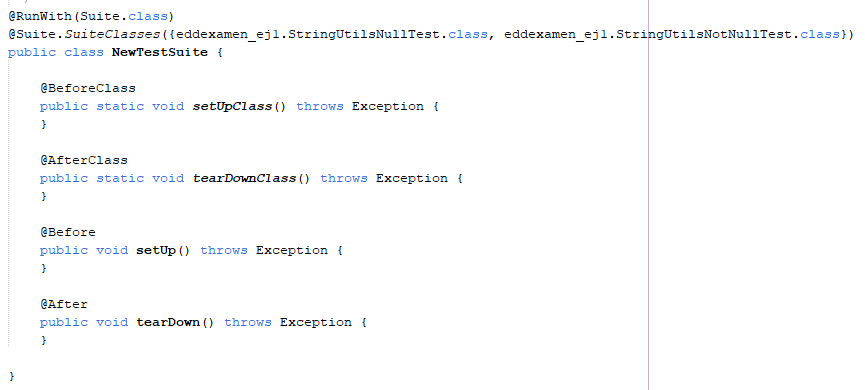
Description automatically generated

PEGA AQUÍ CAPTURA CON EL CÓDIGO DEL APARTADO II

A computer screen shot of text

Description automatically generated

PEGA AQUÍ CAPTURA CON EL CÓDIGO DEL APARTADO III



# ENUNCIADO SEGUNDO EJERCICIO (4 PUNTOS)

Realiza el siguiente diagrama de clases con el uso de la herramienta **Visual Paradigm**. No es necesario indicar getters/setters.

Texto

Descripción generada automáticamente

PEGA AQUÍ LA CAPTURA CON EL DIAGRAMA

A diagram of a computer

Description automatically generated

# ENUNCIADO TERCER EJERCICIO (2 PUNTOS)

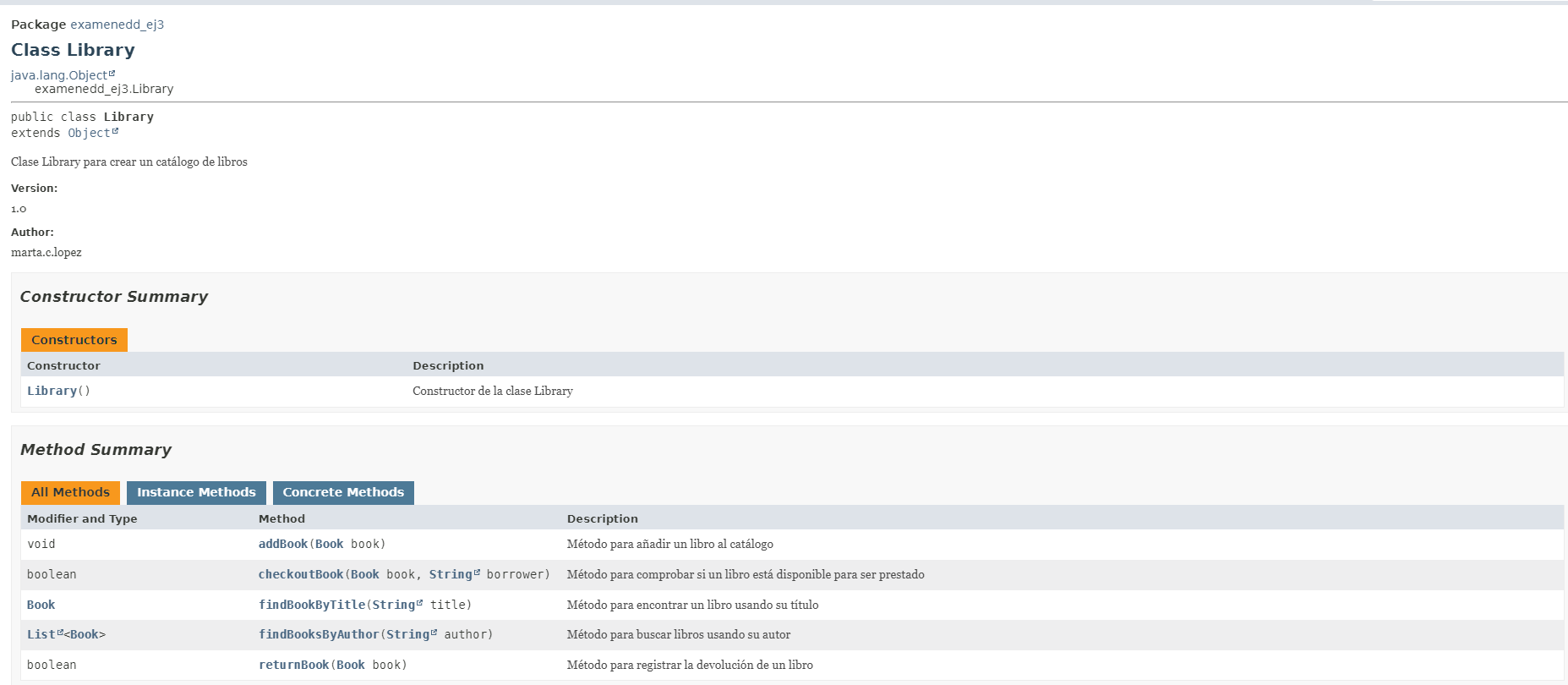
Tenemos el siguiente código JAVA. Realiza la documentación con la herramienta Javadoc. Documenta de **forma detallada** qué realiza el código.

**Comprime** todo la carpeta JAVADOC en una carpeta y **pega capturas** de los archivos HTML al finalizar.

|  |
| --- |
| import java.util.ArrayList;  import java.util.List;  public class Library {  private List<Book> catalog;  public Library() {  this.catalog = new ArrayList<>();  }  public void addBook(Book book) {  catalog.add(book);  }  public Book findBookByTitle(String title) {  for (Book book : catalog) {  if (book.getTitle().equalsIgnoreCase(title)) {  return book;  }  }  return null;  }  public List<Book> findBooksByAuthor(String author) {  List<Book> booksByAuthor = new ArrayList<>();  for (Book book : catalog) {  if (book.getAuthor().equalsIgnoreCase(author)) {  booksByAuthor.add(book);  }  }  return booksByAuthor;  }  public boolean checkoutBook(Book book, String borrower) {  if (book.isCheckedOut()) {  return false; // Book is already checked out  }  book.setCheckedOut(true);  book.setBorrower(borrower);  return true;  }  public boolean returnBook(Book book) {  if (!book.isCheckedOut()) {  return false; // Book is not currently checked out  }  book.setCheckedOut(false);  book.setBorrower(null);  return true;  }  }  class Book {  private String title;  private String author;  private boolean checkedOut;  private String borrower;  public Book(String title, String author) {  this.title = title;  this.author = author;  this.checkedOut = false;  this.borrower = null;  }  public String getTitle() {  return title;  }  public String getAuthor() {  return author;  }  public boolean isCheckedOut() {  return checkedOut;  }  public void setCheckedOut(boolean checkedOut) {  this.checkedOut = checkedOut;  }  public String getBorrower() {  return borrower;  }  public void setBorrower(String borrower) {  this.borrower = borrower;  }  } |

A screenshot of a computer

Description automatically generated



A screenshot of a web page

Description automatically generated

A screenshot of a web page

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated