GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv
CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

EXAMEN PRÁCTICO

ENTORNOS DE DESARROLLO

2ª EVALUACIÓN

CURSO: 2023/2024

RAÚL PALAO LOZANO

1. 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();
    }
}
```

a. Realiza una clase **StringUtilsNullTest** y otra **StringUtilsNotNullTest**.
   i. En la primera deberás probar la función **usando nulo.** (1.5 puntos).
   ii. 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).
   iii. 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:

Apartado I

```java
package ed;

import static org.junit.Assert.assertEquals;
import org.junit.Test;

/**
 *
 * @author Roldán Sanchis Martínez
 */
public class StringUtilsNullTest {

    @Test
    public void testNullReverse() {
        System.out.println(x: "reverse");
        String str = null;
        String expResult = null;
        String result = StringUtils.reverse(str);
        assertEquals(expected: expResult, actual: result);
    }
}
```

Apartado II

```java
package ed;

import java.util.Arrays;
import java.util.Collection;
import org.junit.Test;
import static org.junit.Assert.*;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;

/**
 *
 * @author Roldán Sanchis Martínez
 */
@RunWith(Parameterized.class)
public class StringUtilsNotNullTest {

    private String in;
    private String out;

    public StringUtilsNotNullTest(String in, String out) {
        this.in = in;
        this.out = out;

    }

    @Parameterized.Parameters
    public static Collection<Object[]> entradas() {
        return Arrays.asList(new Object[][]{
            {"asd "," dsa"},
            {"Mayus", "suyaM"},
            {"caña","añac"},
            {"calçot","toçlac"},
            {"Roldán", "nádloR"}
        });
    }

    @Test
    public void testNotNullReverse() {
        System.out.println(x: "reverse");
        String str = this.in;
        String expResult = this.out;
        String result = StringUtils.reverse(str);
        assertEquals(expected: expResult, actual: result);
    }

}
```

Apartado III

```java
package suites;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

/**
 *
 * @author Roldán Sanchis Martínez
 */
@RunWith(Suite.class)
@Suite.SuiteClasses({ed.StringUtilsNotNullTest.class,ed.StringUtilsNullTest.class})
public class StringUtilsTestSuite {

}
```

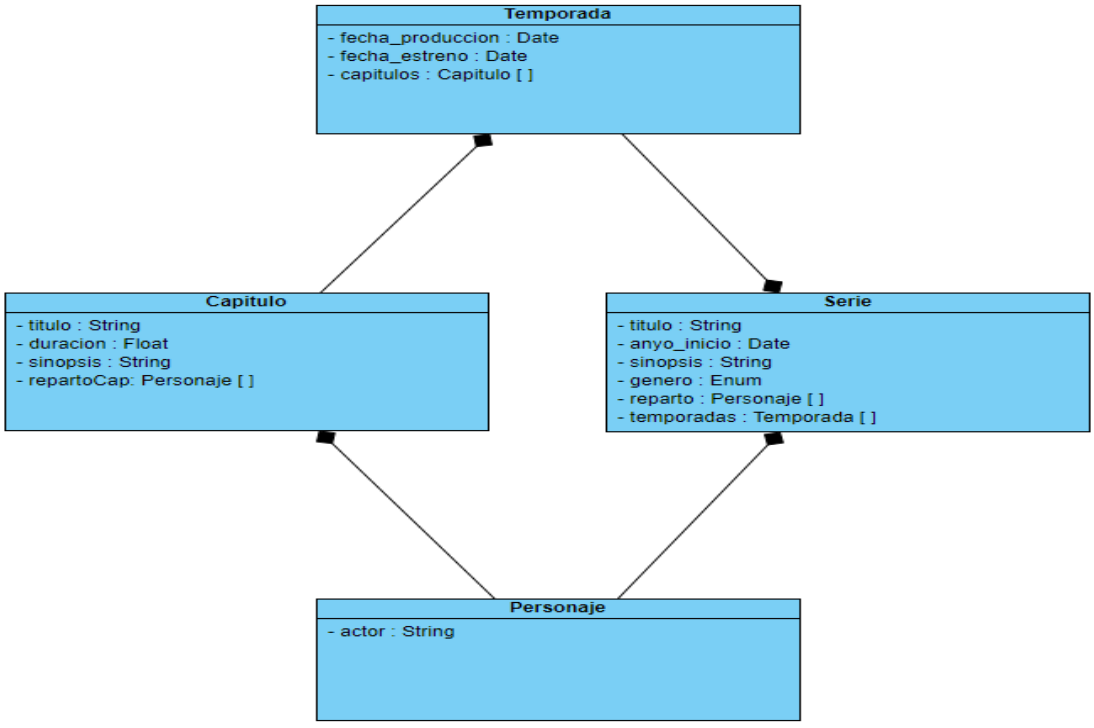
2. ENUNCIADO SEGUNDO EJERCICIO (4 PUNTOS)

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

	ENUNCIADO

Representa mediante un diagrama de clases la siguiente especificación relacionada con un sistema para gestionar series

- Las series se caracterizan por su título, año de inicio, sinopsis, género al que pertenece (acción, aventura, animación, comedia, documental, drama, horror, musical, romance, ciencia ficción) y personajes que intervienen.
- Las series se organizan en temporadas ordenadas que tienen una fecha de producción y una fecha de estreno de televisión a nivel mundial.
- Cada temporada está a su vez formada por capítulos ordenados que tienen un título, una duración y una sinopsis.
- Un personaje en una serie concreta es interpretado por un único actor pero un actor puede interpretar varios personajes en una misma serie.
- Un personaje interpretado por un actor puede aparecer en más de una serie.
- Además un personaje puede no aparecer en todos los capítulos de la serie por lo que el sistema debe conocer en qué capítulos aparece un personaje.



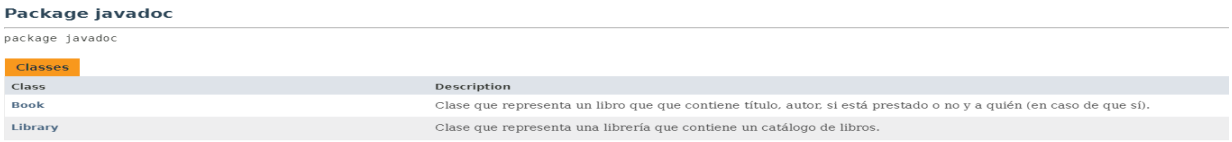
3. ENUNCIADO TERCER EJERCICIO (2 PUNTOS)

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;
    }
}
```

Javadoc general:



Package javadoc

package javadoc

Classes

Class	Description
Book	Clase que representa un libro que que contiene título, autor, si está prestado o no y a quién (en caso de que sí).
Library	Clase que representa una librería que contiene un catálogo de libros.

Javadoc Library:

Package javadoc

Class Library

java.lang.Object
  javadoc.Library

public class **Library**
extends Object

Clase que representa una librería que contiene un catálogo de libros.

Version:
1.0 07-05-2024

Author:
Raúl Palao *feat.* **Roldán Sanchis Martínez**

Field Summary

Fields

Modifier and Type	Field	Description
private List<Book>	catalog	Catálogo de libros

Constructor Summary

Constructors

Constructor	Description
Library()	Constructor vacío de la clase Library.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	addBook(Book book)	Añade un libro al catálogo de libros.
boolean	checkoutBook(Book book, String borrower)	Intenta prestar un libro a un usuario y dice si ha sido posible el préstamo.
Book	findBookByTitle(String title)	Busca un libro en el catálogo de libros dado su título y lo devuelve.
List<Book>	findBooksByAuthor(String author)	Busca un conjunto de libros por su autor y los retorna en forma de lista.
boolean	returnBook(Book book)	Intenta devolver un libro a la librería y dice si ha sido posible o no.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Details

catalog

private List<Book> catalog

Catálogo de libros

Constructor Details

Library

public Library()

Constructor vacío de la clase Library. Únicamente añade un catálogo vacío de libros.

Method Details

addBook

public void addBook(Book book)

Añade un libro al catálogo de libros.

Parameters:
book - El libro que se va a añadir.

findBookByTitle

public Book findBookByTitle(String title)

Busca un libro en el catálogo de libros dado su título y lo devuelve.

Parameters:
title - Título del libro a buscar.

Returns:
Devuelve un objeto de la clase Book dado un título. Si el título no está entre los títulos de los libros devuelve NULL.

Constructor Details

Library

```
public Library()
```

Constructor vacío de la clase Library. Únicamente añade un catálogo vacío de libros.

Method Details

addBook

```
public void addBook(Book book)
```

Añade un libro al catálogo de libros.

Parameters:
`book` - El libro que se va a añadir.

findBookByTitle

```
public Book findBookByTitle(String title)
```

Busca un libro en el catálogo de libros dado su título y lo devuelve.

Parameters:
`title` - Título del libro a buscar.

Returns:
Devuelve un objeto de la clase Book dado un título. Si el título no está entre los títulos de los libros devuelve NULL.

findBooksByAuthor

```
public List<Book> findBooksByAuthor(String author)
```

Busca un conjunto de libros por su autor y los retorna en forma de lista.

Parameters:
`author` - Nombre del autor de los libros a buscar.

Returns:
Devuelve una List de Book con los libros del autor. Si no hay libros de un autor devuelve una lista vacía.

checkoutBook

```
public boolean checkoutBook(Book book,
                            String borrower)
```

Intenta prestar un libro a un usuario y dice si ha sido posible el préstamo.

Parameters:
`book` - El libro a prestar.

`borrower` - El nombre del usuario del préstamo.

Returns:
Devuelve un booleano dependiendo de si ha sido posible o no el préstamo:
- True : Se ha realizado el préstamo correctamente.
- False : El préstamo ya se había realizado (puede que a otro usuario).

returnBook

```
public boolean returnBook(Book book)
```

Intenta devolver un libro a la librería y dice si ha sido posible o no.

Parameters:
`book` - Libro a devolver.

Returns:
Devuelve un booleano dependiendo de si se ha podido devolver o no.
- True : El libro se ha podido devolver correctamente.
- False : el libro no se ha podido devolver (no estaba prestado).

Javadoc Book:

Package javadoc

Class Book

java.lang.Object
javadoc.Book

class Book
extends Object

Clase que representa un libro que que contiene título, autor, si está prestado o no y a quién (en caso de que sí).

Version:
1.0 07-05-2024

Author:
Raúl Palao *feat.* Roldán Sanchis Martínez

Field Summary

Fields

Modifier and Type	Field	Description
private String	author	Autor del libro
private String	borrower	Usuario del préstamo.
private boolean	checkedOut	Si el libro está prestado o no.
private String	title	Título del libro

Constructor Summary

Constructors

Constructor	Description
Book(String title, String author)	Constructor de la clase Book con parámetros.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
String	getAuthor()	Devuelve el autor del libro.
String	getBorrower()	Devuelve a quién se ha prestado el libro.
String	getTitle()	Devuelve el título del libro.
boolean	isCheckedOut()	Dice si el libro está prestado o no.
void	setBorrower(String borrower)	Actualiza el nombre del usuario del préstamo.
void	setCheckedOut(boolean checkedOut)	Actualiza el valor de prestado: True : está prestado. False : no está prestado.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Details

title

private String title

Título del libro

author

private String author

Autor del libro

checkedOut

private boolean checkedOut

Si el libro está prestado o no.

borrower

private String borrower

Usuario del préstamo.

Constructor Details

Book

```
public Book(String title,
            String author)
```

Constructor de la clase Book con parámetros. Crea un libro con título y autor, pero sin préstamo (checkedOut = false y borrower = null).

Method Details

getTitle

public String getTitle()

Devuelve el título del libro.

getAuthor

public String getAuthor()

Devuelve el autor del libro.

isCheckedOut

```
public boolean isCheckedOut()
```

Dice si el libro está prestado o no.

setCheckedOut

```
public void setCheckedOut(boolean checkedOut)
```

Actualiza el valor de prestado:

- True : está prestado.
- False : no está prestado.

getBorrower

```
public String getBorrower()
```

Devuelve a quién se ha prestado el libro.

setBorrower

```
public void setBorrower(String borrower)
```

Actualiza el nombre del usuario del préstamo. Si es null quiere decir que no está prestado.