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:

PEGA AQUÍ CAPTURA CON EL CÓDIGO DEL APARTADO I

```
/**
 * Test of reverse method, of class StringUtils.
 */
@Test
public void testReverse() {
    System.out.println(x: "reverse");
    StringUtils instance = new StringUtils();
    assertNull(object: instance.reverse(str:null));
}
```

PEGA AQUÍ CAPTURA CON EL CÓDIGO DEL APARTADO II

```
@Parameters
public static Collection<Object[]> valores(){
    return Arrays.asList(new Object[][]{{"HoLa", "aLoH"}, {"hola que tal", "lat euq aloh"},
        {"cañada", "adañac"},{"pastel", "letsap"},{"buenos dias", "said soneub"}});
}

/**
 * Test of reverse method, of class StringUtils.
 */
@Test
public void testReverse() {
    System.out.println(x: "reverse");
    StringUtils instance = new StringUtils();
    String resReal = instance.reverse(str:this.entrada);
    String resEsperado = this.resultado;
    assertEquals(expected: resEsperado, actual: resReal);
}
```

PEGA AQUÍ CAPTURA CON EL CÓDIGO DEL APARTADO III

```
@RunWith(Suite.class)
@Suite.SuiteClasses({eddexamen_ej1.StringUtilsNullTest.class, eddexamen_ej1.StringUtilsNotNullTest.class})
public class NewTestSuite {

    @BeforeClass
    public static void setUpClass() throws Exception {
    }

    @AfterClass
    public static void tearDownClass() throws Exception {
    }

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

}
```

2. ENUNCIADO SEGUNDO EJERCICIO (4 PUNTOS)

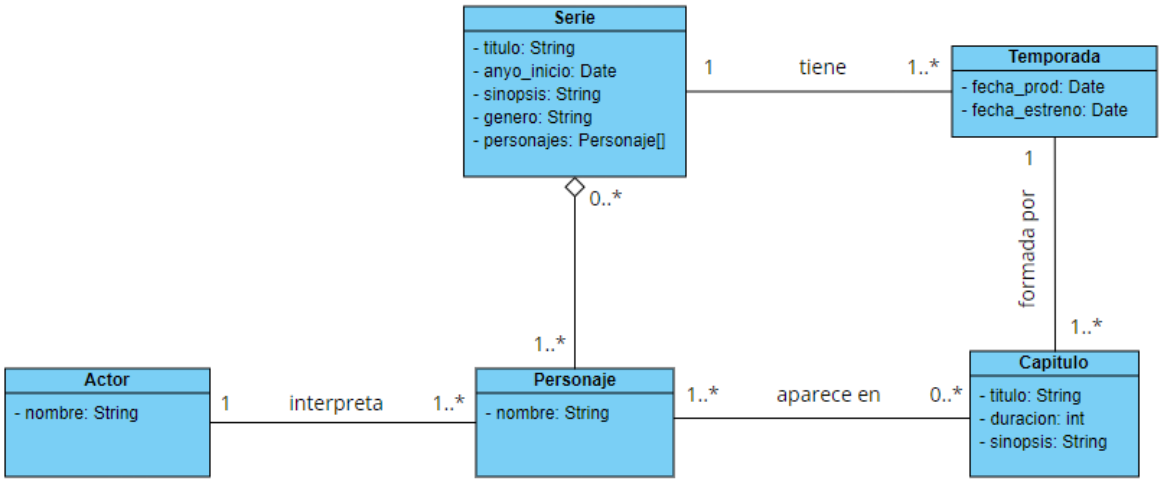
Realiza el siguiente diagrama de clases con el uso de la herramienta **Visual Paradigm**. No es necesario indicar getters/setters.

	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.

PEGA AQUÍ LA CAPTURA CON EL DIAGRAMA

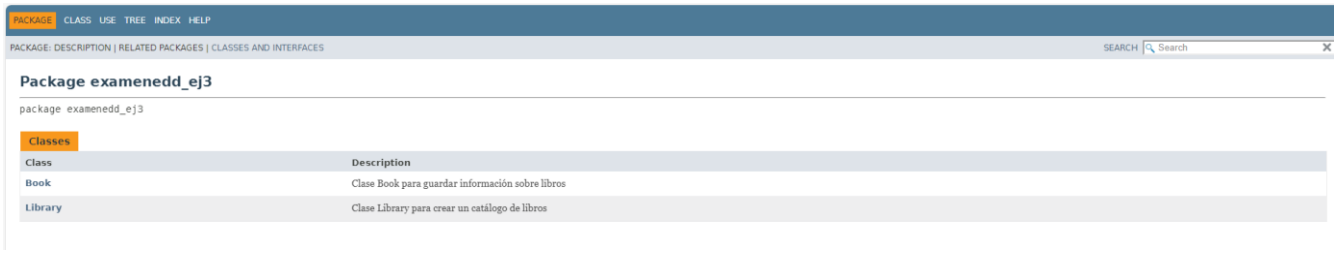


3. ENUNCIADO TERCER EJERCICIO (2 PUNTOS)

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;
    }
}
```



PACKAGE CLASS USE TREE INDEX HELP

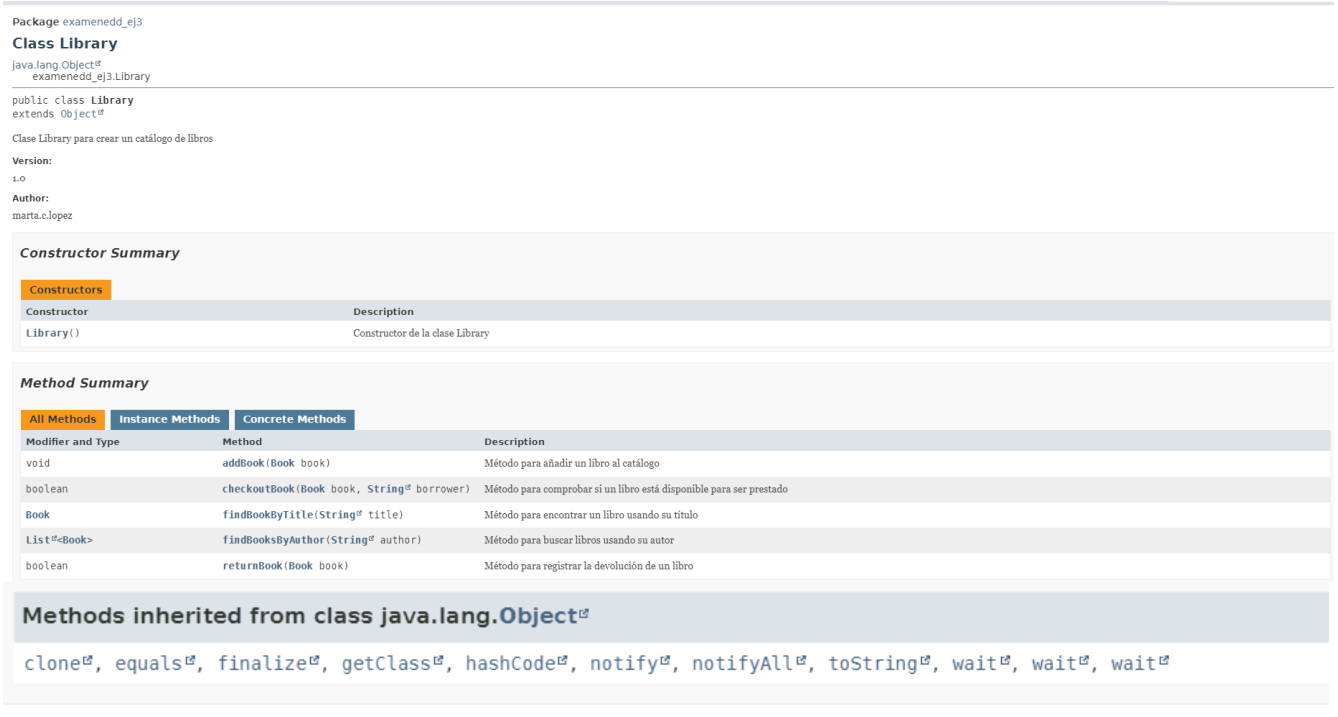
PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES SEARCH Search

Package examenedd_ej3

package examenedd_ej3

Classes

Class	Description
Book	Clase Book para guardar información sobre libros
Library	Clase Library para crear un catálogo de libros



Package examenedd_ej3

Class Library

java.lang.Object
examenedd_ej3.Library

public class Library
extends Object

Clase Library para crear un catálogo de libros

Version:

1.0

Author:

marta.c.lopez

Constructor Summary

Constructors

Constructor	Description
Library()	Constructor de la clase Library

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	addBook(Book book)	Método para añadir un libro al catálogo
boolean	checkoutBook(Book book, String borrower)	Método para comprobar si un libro está disponible para ser prestado
Book	findBookByTitle(String title)	Método para encontrar un libro usando su título
List<Book>	findBooksByAuthor(String author)	Método para buscar libros usando su autor
boolean	returnBook(Book book)	Método para registrar la devolución de un libro

Methods inherited from class java.lang.Object

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

Constructor Details

Library

```
public Library()
```

Constructor de la clase Library

Method Details

addBook

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

Método para añadir un libro al catálogo

Parameters:

book - libro que se añade al catálogo

findBookByTitle

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

Método para encontrar un libro usando su título

Parameters:
`title` - título del libro que buscamos en el catálogo

Returns:
libro que coincide con el título buscado

findBooksByAuthor

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

Método para buscar libros usando su autor

Parameters:
`author` - autor del libro que queremos encontrar

Returns:
lista de libros escritos por un determinado autor

checkoutBook

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

Método para comprobar si un libro está disponible para ser prestado

Parameters:
`book` - libros que se va a prestar

`borrower` - persona que se lleva el libro

Returns:
falso si el libro está prestado, verdadero si el libro está disponible y se presta ahora

returnBook

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

Método para registrar la devolución de un libro

Parameters:
`book` - libro que se va a devolver

Returns:
falso si el libro no está prestado, verdadero si el libro está prestado y se devuelve.

Package examenedd_ej3

Class Book

java.lang.Object
examenedd_ej3.Book

public class **Book**
extends Object

Clase Book para guardar información sobre libros

Version:

1.0

Author:

marta.c.lopez

Constructor Summary

Constructors

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

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
String	getAuthor()	Método getter para obtener el autor del libro
String	getBorrower()	Método getter para obtener la persona que ha sacado el libro prestado.
String	getTitle()	Método getter para obtener el título del libro
boolean	isCheckedOut()	Método getter para obtener el valor del atributo isCheckedOut, para comprobar si un libro está prestado o no.
void	setBorrower(String borrower)	Método getter para asignar un valor al atributo borrower
void	setCheckedOut(boolean checkedOut)	Método setter para asignar un valor al atributo isCheckedOut.

Methods inherited from class java.lang.Object

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

Constructor Details

Book

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

Constructor de la clase Book

Parameters:

`title` - titulo del libro

`author` - autor del libro

Method Details

getTitle

```
public String getTitle()
```

Método getter para obtener el título del libro

Returns:

título del libro

getAuthor

`public String getAuthor()`

Método getter para obtener el autor del libro

Returns:
autor del libro

isCheckedOut

`public boolean isCheckedOut()`

Método getter para obtener el valor del atributo isCheckedOut, para comprobar si un libro está prestado o no.

Returns:
valor del atributo isCheckedOut

setCheckedOut

`public void setCheckedOut(boolean checkedOut)`

Método setter para asignar un valor al atributo isCheckedOut.

Parameters:
`checkedOut` - valor booleano para indicar si el libro está prestado o no

getBorrower

`public String getBorrower()`

Método getter para obtener la persona que ha sacado el libro prestado.

Returns:
nombre de la persona que ha sacado un libro prestado.

setBorrower

`public void setBorrower(String borrower)`

Método getter para asignar un valor al atributo borrower

Parameters:
`borrower` - nombre de la persona que saca un libro prestado.