Java Programming 1

Project assignment

Learning outcomes							ALL
LO1	LO2	LO3	LO4	LO5	LO6	LO7	
15	15	15	15	15	15	10	100

INSTRUCTIONS

- The defense of the project takes place during the examination periods
- The student applies as well as for written exams

LEARNING OUTCOMES:

LO1 (15 points):

- *minimum (10 points):* Implement the basic concepts of object-oriented paradigm in object-oriented programming language on a virtual platform
- *desired (5 points)*: Implement advanced concepts of object-oriented paradigm in object-oriented programming language on a virtual platform

LO2 (15 points):

- *minimum (10 points):* Apply a functional paradigm and evaluate it in relation to an object-oriented paradigm
- *desired (5 points)*: When implementing an application, select and apply an appropriate functional or object-oriented paradigm

LO3 (15 points):

- *minimum (10 points):* Identify the need to use the collection framework and flows and implement them according to best practices
- *desired (5 points)*: When deploying the application, use advanced implementations of the collection framework and flows

LO4 (15 points):

- *minimum (10 points):* Compare and evaluate the traditional and modern way of working with the data system
- *desired (5 points)*: Implement a modern way of working with the data system

LO5 (15 points):

- *minimum (10 points):* Apply appropriate libraries to design standard graphics software solutions
- *desired (5 points)*: Apply appropriate libraries to design advanced graphics software solutions

LO6 (15 points):

- *minimum (10 points):* Apply appropriate libraries based on MVC architecture in designing standard graphical software solutions
- *desired (5 points)*: Apply appropriate libraries based on MVC architecture in the design of advanced graphics software solutions

LO7 (10 points):

- *minimum (7 points):* Compare different approaches when creating graphic software solutions
- *desired (3 points)*: Use advanced approaches when creating graphical software solutions

Create an application that allows updating data about connected entities, as desired. In the following, an example of the **Movie** entity and associated entities **Actor**, **Director, Genre** and the like will be explained.

When building the application, it is important to respect the best principles of the object-oriented paradigm and using libraries (**Class Library**), according to the following instructions:

- The application saves data in the **Microsoft SQLServer** database
- It is necessary to create an initialization **DDL** script to create all the tables that will be used in the application
- It is necessary to create a script to delete all data from the tables
- All work with the database takes place by calling procedures from Java code, using the **Repository** pattern
- The application contains several types of users, so it is necessary to provide 2 roles - **Administrator** and **User**
- The **Administrator** must be at least one user with the associated username/password pair that will be created using the procedure after the initialization script
- Access to the application is restricted to the **Login** form, where simple user registration of the **User role** is also possible
- The application is put in the correct state for use by the **User** by the **Administrator**
    - When entering the application, the **Administrator** has the option of deleting all data from the database (which also deletes all images from the data system) and uploading new data to the database, by calling the **RSS parser** - this represents a simple and complete administrator interface
- **RSS parser** is an application component that parses all XML data from a specific URL and saves it in the database for later use
    - Images must be downloaded to a local directory (example: **assets**) and their relative paths must be saved in the database
    - Examples of RSS feeds:
        - https://www.nasa.gov/rss-feeds/
        - https://photojournal.jpl.nasa.gov/rss/index.html
        - https://www.cedefop.europa.eu/en/news-and-events/rss-feeds
        - https://news.sky.com/info/rss
        - https://www.sciencedaily.com/newsfeeds.htm
- After entering the application, the **User** is presented with a form for viewing and changing entities (**CRUD** operations), in this example **Movie**
- Forms within the application must be well organized (example: **JTabbedPane**, where each **JPanel** is represented by its own class according to the **Single-responsibility principle**)
- It is necessary to create additional entities (example: **Actor**, **Director**) that also have **CRUD** forms
- The existing entity needs to be updated in such a way that it can be connected to new entities (example: **Movie** can be connected to several **Actor**, **Director, Genre** entities)
- When changing an entity or deleting it, it is necessary to ensure a consistent change / deletion of the associated images from the image directory (example: **assets**)
- To display entities, it is necessary to use the **JTable** and associated **AbstractTableModel** view-models to the greatest extent
- To navigate in the application, it is necessary to use **JMenu**
- It is necessary to implement **Drag and drop** functionality on your own selected example in the application (example: adding **Actor** to **Movie**)
- It is necessary to implement the **XML download** of any entity using the **JAXB library**

When developing the application, it is important to respect advanced principles of the object-oriented paradigm (SOC - separation of concerns, SRP - single responsibility principle, loose coupling, high cohesion, prefer composition over inheritance, DRY - don't repeat yourself…)

The points are scored according to the following scheme:

LO1 (15 points):

- (***Minimum – 10 points***)
  - The application must demonstrate thorough understanding of basic OOP principles – *Encapsulation*, *Inheritance*, *Polymorphism* and *Abstractions* in the example of connected entities that form the architecture (in the example – Movie, Actor, Director, Genre, User, Administrator…)
  - Data management and storage must be provided through the interfaces
  - Data entities in the application must be properly encapsulated, implement *equals()*, *hashcode()* and *toString()* methods, as well as *Comparable* interface, where necessary
  - The application must demonstrate thorough understanding of Exception management (***checked*** vs. ***runtime*** exceptions)
  - Utilities in the application must be provides as static classes, using private constructors for instance control
  - Usage of properly encapsulated class libraries as Maven modules for ***Utilities*** and ***Dao*** is highly recommended
- (***Desired – 5 points)***
  - Application should use Singleton (Lazy/Eager) and Repository patterns
  - Enum should be used in order to create statically typed RSS parser
  - Generics should be used to prevent redundancies

LO2 (15 points):

- (***Minimum – 10 points***)
  - The application must demonstrate thorough understanding of functional paradigm in relation to an object-oriented paradigm – will be discussed on project defense
  - Application must demonstrate usage of
    - ***Consumer*** functional interface
    - ***Predicate*** functional interface
    - ***Function*** functional interface
    - ***Filters***
    - ***Optional*** implemented through the functional paradigm
    - ***default methods***
- (***Desired – 5 points)***
  - During defense, demonstrate the usage functional paradigm in regards to object-oriented paradigm
  - Explain pros and cons of functional paradigm vs object-oriented paradigm using concrete examples in the application

LO3 (15 points):

- (***Minimum – 10 points***)
  - The application must demonstrate thorough understanding of *Collections framework*
  - The application must demonstrate the paradigm ***program to abstractions, not implementations***
  - Use polymorphic solutions for the validations, using ***List*** or ***Map*** interfaces
  - Demonstrate the usage of ***Set*** interface to prevent duplication
  - During defense, explain the *Collections framework* hierarchy and identify the scenarios for the proper usage of the following implementations
    - ***ArrayList*** vs. ***LinkedList***
    - ***HashSet*** vs. ***TreeSet***
    - ***HashMap*** vs. ***TreeMap***
- (***Desired – 5 points***)
  - The application should demonstrate advanced usage Collections framework utilities
  - Use ***distinct()***, ***sorted()***, ***skip()***, ***min()***/***max()***, ***findAny()***/***findFirst()***, **anyMatch()/noneMatch()** and **Collectors** to demonstrate advanced usage of ***Streams***

LO4 (15 points):

- (***Minimum – 10 points***)
  - Create an initialization DDL script to create all the tables that will be used in the application
  - Create a script to delete all data from the tables
  - The data of the application must be stored in the database, respecting the best principles
  - Initial data must be downloaded from RSS source, using ***XML parser*** and ***HttpUrlConnection,*** and then stored in the database
  - CRUD must be implemented by calling procedures from Java code, using the **Repository pattern**
  - During project defense compare and evaluate usage CRUD queries with and without procedures
- (***Desired – 5 points***)
  - It is necessary to implement the ***XML download*** of chosen entity using the ***JAXB library***
  - Usage of ***PreparedStatement***/***CallableStatement*** to prevent sql injection is highly recommended

LO5 (15 points):

- (***Minimum – 10 points***)
  - The application must demonstrate understanding of standard graphics solutions by proper usage of ***Swing Application Framework***
  - Demonstrate understanding of components lifecycle
  - Forms within the application must be well organized
  - Create ***Utilities*** to encapsulate *and simplify usage of*
    - ***JOptionDialog***
    - ***JFileChooser***
  - Use ***JMenu*** to enable user to navigate through the application
  - Implement ***Drag and drop*** functionality on your own selected example in the application
- (***Desired – 5 points***)
  - Use Maven dependency to enable ***FlatLaf*** library
  - Create ***Utility*** *for* Icon management and scaling of images

LO6 (15 points):

- **(*Minimum – 10 points*)**
  - Application must demonstrate thorough understanding of ***Swing MVC*** architecture by proper and appropriate usage of the following **View** entities
    - **List**
    - ***Spinner***
    - ***ComboBox***
    - ***ButtonGroup***
  - **Models** of aforementioned entities must be handled with proper type safety precautions and dynamically loaded from database or enumerated
  - Appropriate event-handling mechanisms must be demonstrated in sense of communicating with **Controller**
- **(*Desired – 5 points*)**
  - To display entities on the screen, it is necessary to use the **JTable**
  - Implement **AbstractTableModel** to enable dynamic loading and sofisticated usage of ***MVC*** patern

LO7 (10 points):

- **(*Minimum – 7 points*)**
  - GUI of the application must be implemented with respect to SOC principle
    - ***JFrame*** must be solely the host or placeholder for the ***JPanel*** instances
    - ***JTabbedPane*** or ***SplitPane*** must be used for organizing ***JPanel*** instances
  - During project defense, thorough understanding ***JFrame*** and ***JPanel*** lifecycles must be demonstrated
  - Application must demonstrate usage of multiple ***JFrame*** instances in proper parent-child relationship
  - Comparison of different ***Swing Layout management*** must be demonstrated on the project defense
- **(*Desired – 3 points*)**
  - The GUI of the application should not be *frozen* during the XML parsing and database loading process
    - Use different thread to parse and load the data
    - Thorough understanding of **Main** and **Event Thread management** of **Swing GUI** should be demonstrated during defense