Weekly Progress Report

Name: Himashu Dhiman

Domain: Core Java

Date of submission: 24-Aug-2024

Final Report:-

1. Overview:

Setting up the Java environment (JDK and IDE) and learning the fundamentals of Java programming are the main goals of the first week. While writing their first "Hello, World!" program, students discover Java's architecture, which includes the main() function. We introduce fundamental ideas like operators, variable declarations, and primitive data types. Students also use the Scanner class and System.out.println() to perform fundamental input/output operations.

The Second week covers Looping constructions and decision-making are covered this week. Students gain knowledge of how to manage program flow using if, else, and switch statements. For recurring tasks, loops (for, while, do-while) are added. Additionally, this week's lesson covers arrays, which teach students how to store and manage data collections. Students will be able to write programs that make use of loops, conditionals, and arrays by the conclusion of the week.

The third week covers the fundamentals of Object-Oriented Programming (OOP), which underpins Java programming. The foundational skills of object-oriented programming (OOP) are taught to students. Additionally, the idea of constructors—which describe how objects are initialized—is discussed.

One of the main OOP concepts, encapsulation, is covered in great detail. In order to maintain data security and integrity, students learn how to employ access modifiers (private, public, and protected) to restrict access to other students in the class. Additionally, they study about the static keyword, which permits fields and methods to be class members as opposed to class instances.

Students should be able to construct classes, instantiate objects, and comprehend how Java's encapsulation and static members function at the end of this week.
Students go further into OOP ideas including inheritance, polymorphism, and abstraction in Week 4. Code reuse is encouraged by inheritance, which lets a new class inherit attributes and functions from an existing class. To access constructors and methods of superclasses, the super keyword is added.

Another important idea is polymorphism, which teaches students about dynamic method dispatch and method overriding, allowing for flexibility in method calls. Students learn how to enforce specific behaviors in subclasses or implement classes by studying abstract classes

and interfaces.

By the end of this week, students ought to know how to use abstract classes and interfaces to construct reusable and flexible code, as well as how to establish a class hierarchy and apply polymorphism.

More complex subjects, including as multithreading, file input/output (I/O), and exception handling, are covered in the last week. Building reliable programs that can elegantly handle runtime problems requires exception handling. Students gain knowledge about creating custom exceptions as well as using the try, catch, and finally blocks.

The introduction of Java file handling enables students to use classes like File, FileReader, and BufferedReader to read from and write to files. The fundamentals of the Java Collections Framework, which offers data structures like ArrayList and HashMap, are also covered during the week.

2. Introduction:

This report details the work completed during the last week of the Java-based Music Player project. The project's goal is to provide a basic music player that users can use to cycle among music tracks and play, pause, and stop it. The player offers a simple user interface for interaction and supports popular audio file types.

3. Objectives:

- Create a Java music player that works.
- Support audio files in WAV and MP3 formats.
- Provide play, pause, stop, and navigation tracking controls.
- For user interaction, put in place a simple graphical user interface (GUI).

4. Technology Stack:

- **Programming Language:** Java
- **Libraries/Frameworks:**
- JavaFX (for GUI development)
- JavaZoom's JLayer (for MP3 decoding)
- javax.sound.sampled (for audio playback)

5. Final Week Tasks:

Throughout the last week, the following assignments were finished:

5.1 GUI Design Completion:

- JavaFX was used to complete the music player interface's layout and design.
- Play, pause, stop, next, and previous track buttons have been added.
- Incorporated a progress bar showing the length of the track and a volume control slider.

5.2 Features for Audio Playback:

- Used the JLayer Player class to implement audio playback for MP3 files.
- Javax.sound's Clip class has been integrated.sampled for playing back WAV files.
- Functionalities like play, pause, and stop enabled.

5.3 Track Management:

- A framework for organizing and navigating several audio tracks was created.
- Added a playlist function that lets users load and listen to several songs in order.
- Shuffle and repeat modes now have additional support.

5.4 Error Correction and Enhancement:

- Error handling has been implemented to handle playback issues and unsupported file types.
- The algorithm has been optimized to enhance playback performance and manage larger audio files.

5.5 Testing and Debugging:

- Conducted thorough testing of the program on several Linux and Windows systems.
- Resolved flaws pertaining to GUI responsiveness and playback problems.
- Confirmed that all significant functions, such as volume control and track navigation, operate as intended.

6. Challenges Faced:

- The integration of external libraries such as JLayer necessitated meticulous dependency management.
- It was difficult to manage numerous audio formats and make sure that they played smoothly on different operating systems.
- Troubleshooting concurrency problems that came up when playing music, particularly when quickly switching songs.

7. Concluding remarks:

The Music Player project was successfully finished during the last week. A completely functional GUI and audio playback mechanism were among the primary functionalities that were implemented. The application runs smoothly on several platforms, and the project achieves its intended goals.

8. Upcoming Improvements:

- Extending the list of supported audio formats to include FLAC and OGG.
- Integrating tools in the app for managing and creating playlists.

- Improving the GUI to make it more contemporary and intuitive.
- Including customizable sound effects and equalizer adjustments.

9. References:

- **Official JavaFX Documentation:** https://openjfx.io/
- **JavaZoom JLayer:** http://www.javazoom.net/javalayer/javalayer.html
- **Oracle's Java Sound API Guide:** https://docs.oracle.com/javase/7/docs/api/javax/sound/sampled/package-summary.html