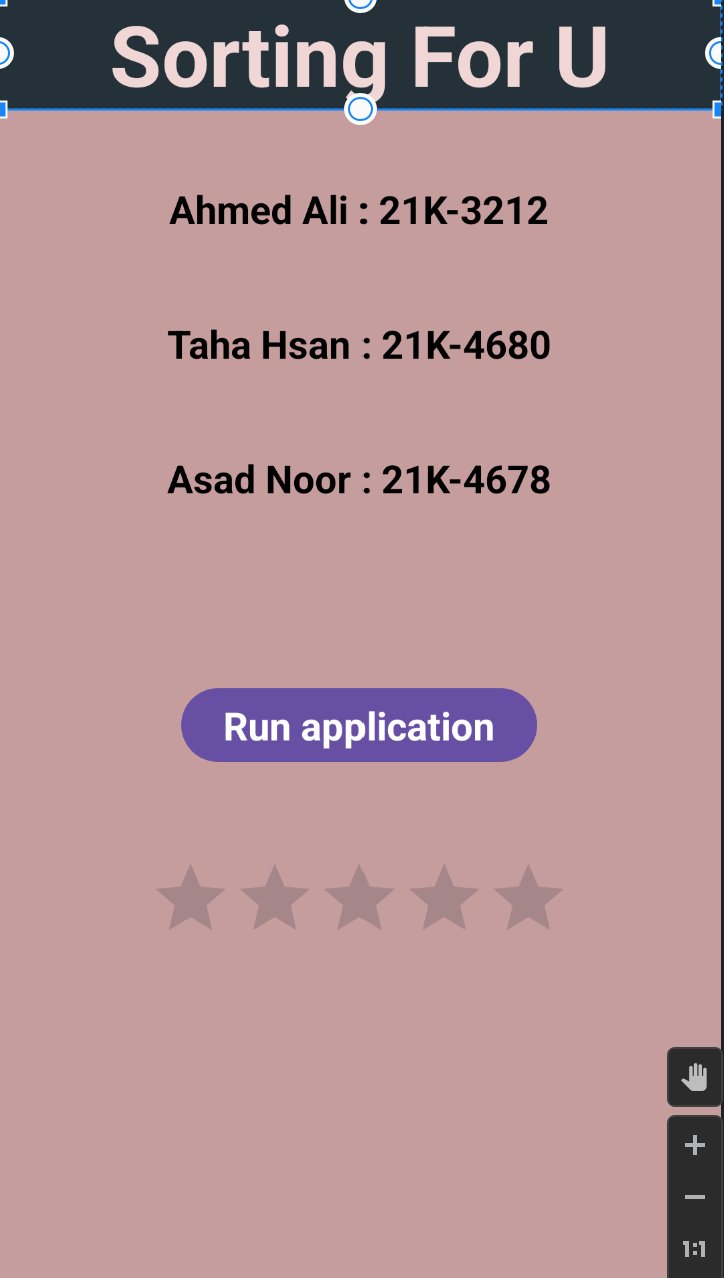
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | | | | |
|  |  | | |  |
| **OS PROJECT REPORT** | | | | |
|  | **Parallel programming in Android – sorting algorithms using threads** | | |  |
| Ahmed ALI (21K-3212)  Taha hassan (21K-4680)  Asad Noor (21k-4678) |  | | |  |

**Introduction:**

The Sorting Algorithm Animation project is an Android application that demonstrates the sorting of an array using three different sorting algorithms: Bubble Sort, Selection Sort, and Insertion Sort. The application provides a visual representation of the sorting process using progress bars and text views.



**Purpose:**

The purpose of this project is to visually illustrate the functioning of different sorting algorithms and provide a better understanding of how they work. It allows users to input an array of integers and see the step-by-step sorting process using animation.

**Technologies Used:**

* Programming Language: Java



* Development Environment: Android Studio

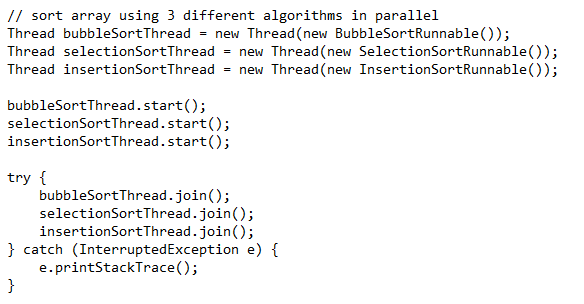


* Android SDK: Android SDK 5.0 (API level 21) and above

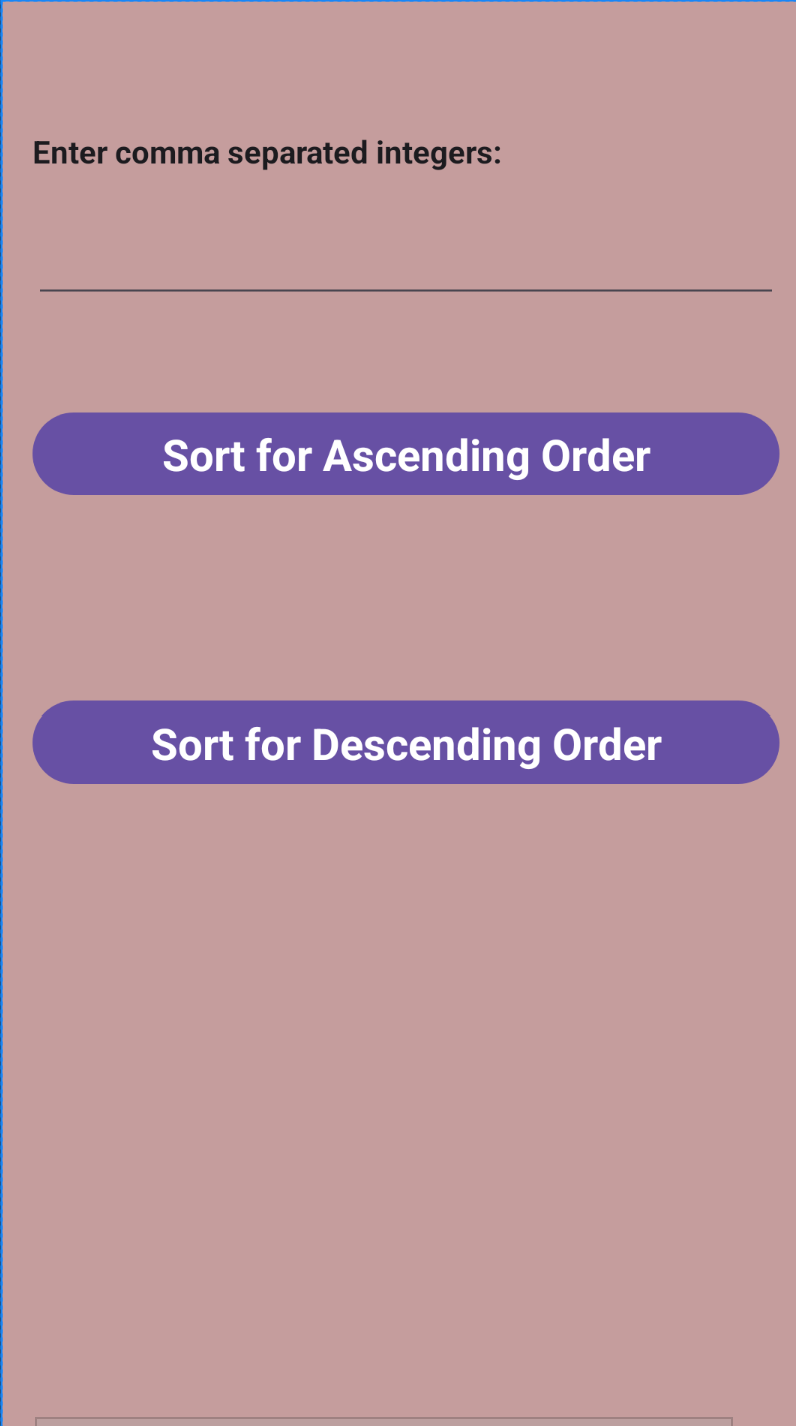


**Features:**

* Visual representation of sorting algorithms: The application provides a visual representation of the sorting process using progress bars and text views.
* Interactive User Interface: Users can input their own array of integers and observe the sorting process in real-time.
* Parallel Execution: The sorting algorithms (Bubble Sort, Selection Sort, and Insertion Sort) are executed in parallel threads to demonstrate their different execution speeds.



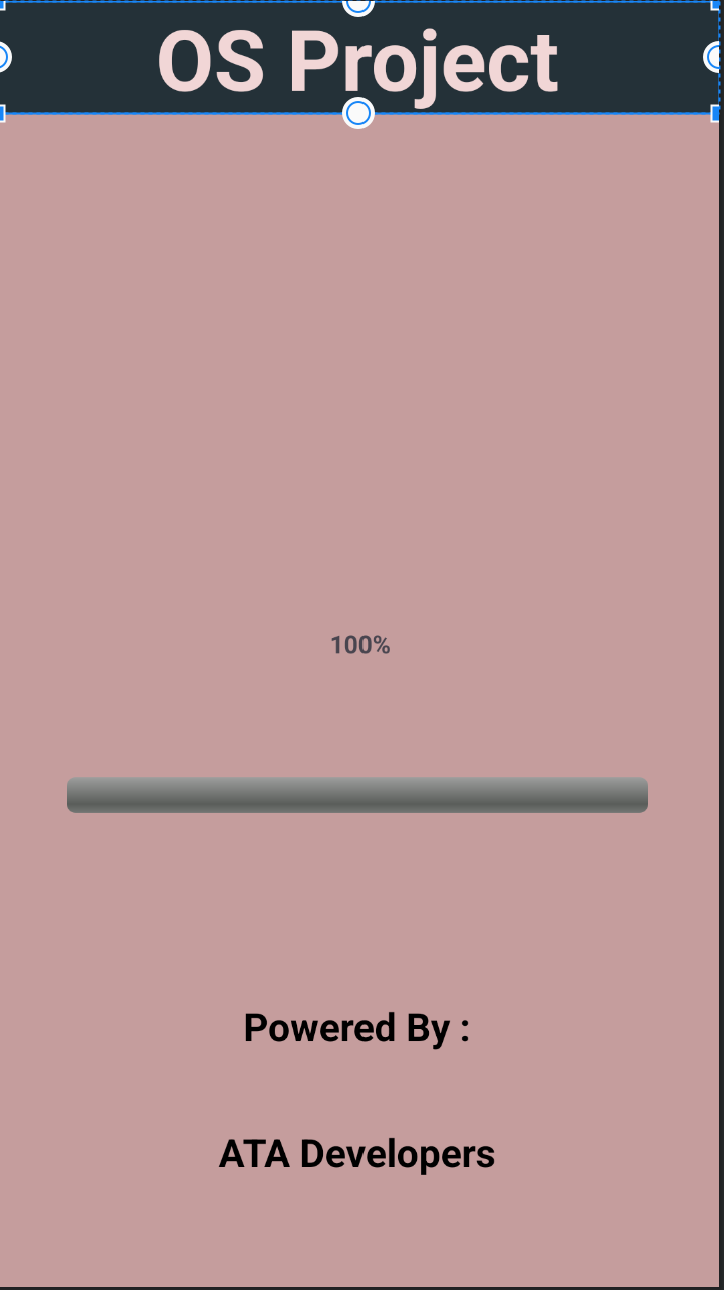
* Ascending and Descending Sorting: Users can choose to sort the array in ascending or descending order.

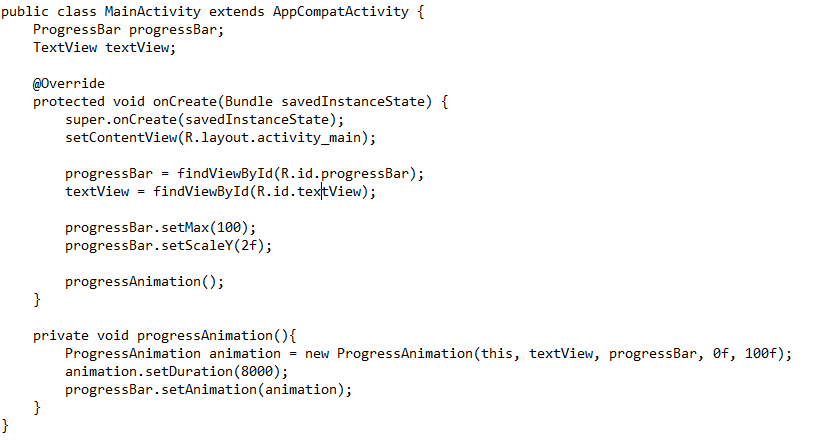


**Code Structure:**

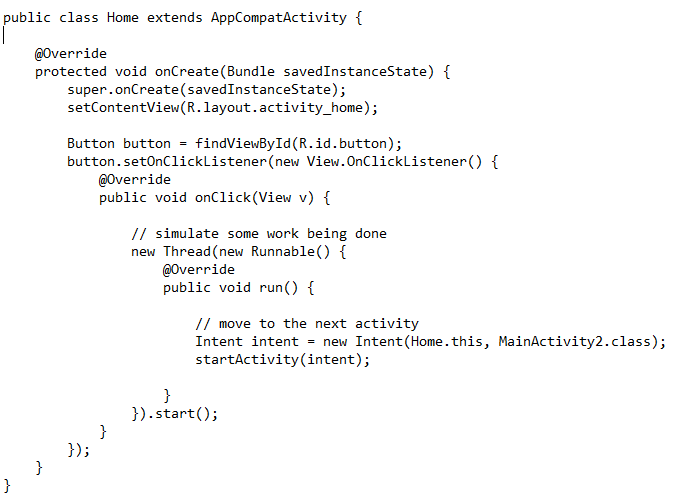
The project consists of three main Java classes and corresponding XML layout files:

**1.)** MainActivity: The initial activity that displays a progress bar and percentage text view to simulate the loading process. It starts the animation and transitions to the Home activity when the progress reaches 100%.

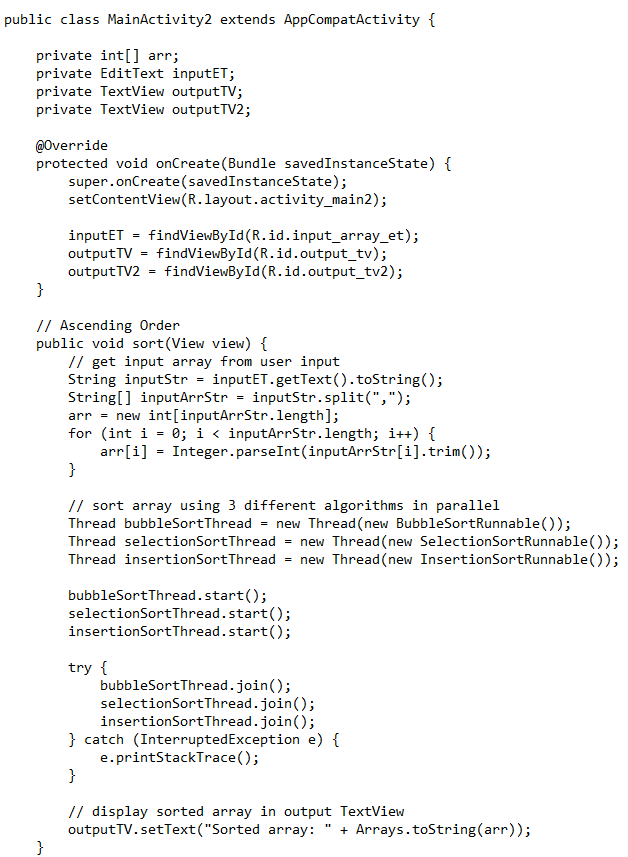


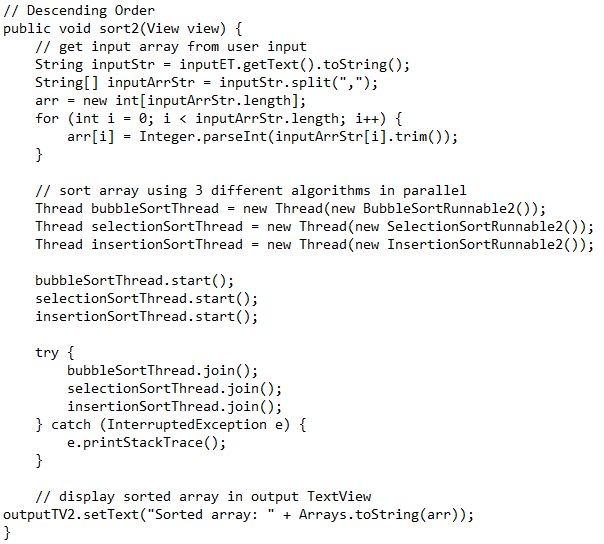


**2.)** HomeActivity: The second activity that provides information about the project and a button to start the sorting process.



**3.)** MainActivity2: The third activity that allows users to input an array of integers and initiates the sorting process using three different sorting algorithms.

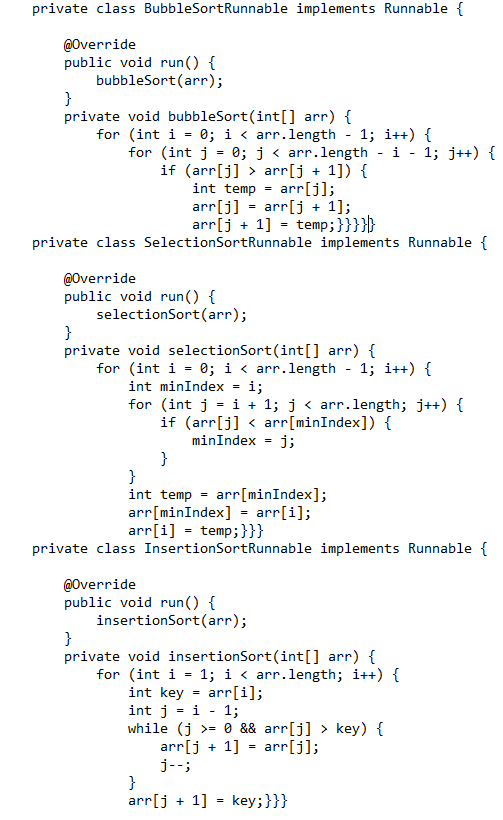




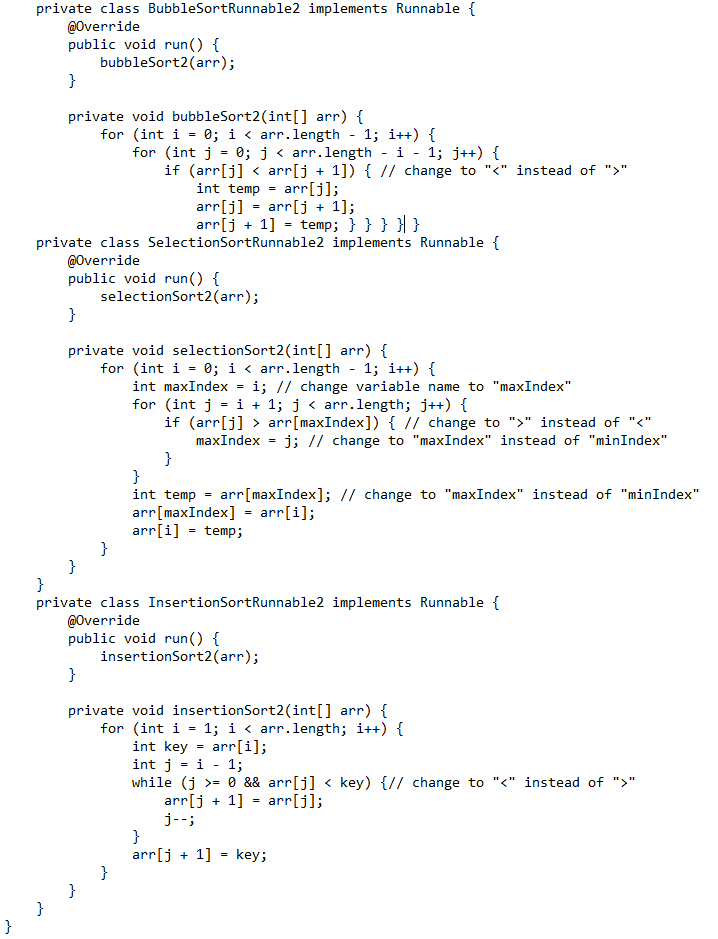
**Sorting Algorithms:**

The sorting algorithms (Bubble Sort, Selection Sort, and Insertion Sort) are implemented as separate classes within the MainActivity2 class. Each sorting algorithm is executed in a parallel thread to demonstrate their different execution speeds. Bubble Sort compares adjacent elements and swaps them if they are in the wrong order. Selection Sort finds the minimum/maximum element and places it in the correct position. Insertion Sort inserts an element into the sorted section of the array.

Ascending sorting by the three algorithms:

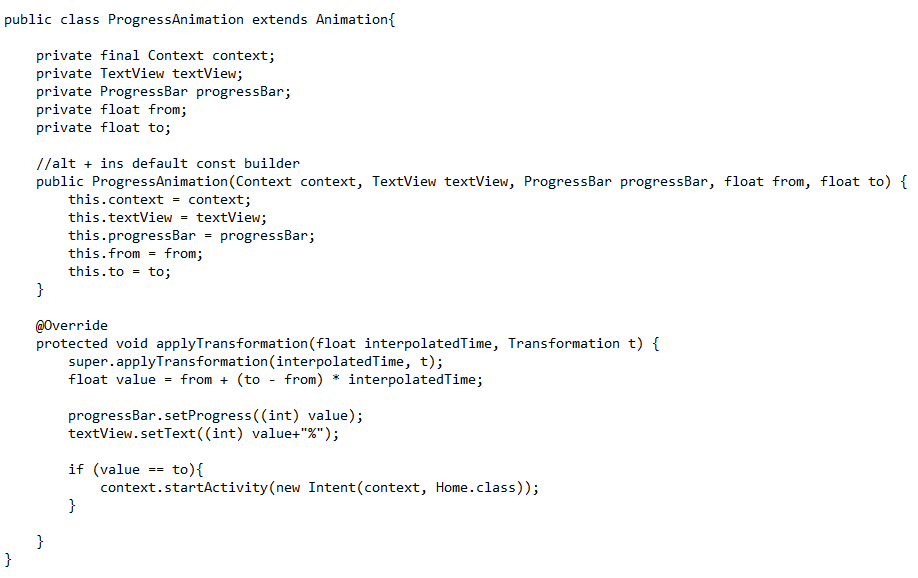


Descending sorting by three algorithms:



**Animation Implementation:**

The ProgressAnimation class extends the Animation class and is responsible for updating the progress bar and text view during the animation. It applies the transformation based on the interpolated time to update the progress bar and text view values. Once the animation reaches its end (value == to), it starts a new activity.



**User Interaction:**

The Home activity allows users to start the sorting process by clicking the "Run application" button. They are then redirected to the MainActivity2 activity, where they can input their array of integers. The application sorts the array using the selected sorting algorithm and displays the sorted array in the output text view.

**Conclusion:**

The Sorting Algorithm Animation project provides an interactive and visual way to understand the functioning of various sorting algorithms. It allows users to observe the step-by-step sorting process and compare the execution speeds of different algorithms. This project can be useful for educational purposes and for anyone interested in learning about sorting algorithms.

**Future Enhancements:**

* Add more sorting algorithms: Expand the project by including additional sorting algorithms such as Merge Sort, Quick Sort, or Heap Sort.
* Improve user interface: Enhance the user interface with more appealing visuals, animations, and user-friendly features.
* Optimization and performance improvements: Optimize the sorting algorithms for better performance and efficiency.
* Error handling and input validation: Implement error handling and input validation to handle incorrect input from the user.