```
package a9;

 import java.util.ArrayList;

 import java.util.HashMap;

 public class PageReplacement {

    static int pageFrames = 0;

    // Least Recently Used (LRU) Page Replacement Algorithm

    static int lru(int referenceString[]) {

        // This array list will contain all the pages that are currently in memory

        ArrayList<Integer> pages = new ArrayList<Integer>(pageFrames);

        // This hashmap will store least recently used indexes of the pages

        HashMap<Integer, Integer> indexes = new HashMap<>();

        int page_faults = 0, n = referenceString.length, curPage;

        for (int i = 0; i < n; i++) {

            curPage = referenceString[i];

            // Check if the set can hold more pages

            if (pages.size() < pageFrames) {

                // Insert it into set if not already present

                // This represents a page fault

                if (!pages.contains(curPage)) {

                    pages.add(curPage);

                    // Increment page fault count

                    page_faults++;

                    displayPageFrames(pages, page_faults);

                }
    // Store the recently used index of each page

    indexes.put(curPage, i);

}

// If the set is full, select a page to be replaced

else {

    // Check if the current page is not already present in the set

    if (!pages.contains(curPage)) {

        // The page having the lowest value of associated index will be the least recently used page

        int lru = Integer.MAX_VALUE, pageToBeReplaced = 0;

        int temp;

        for (int j = 0; j < pages.size(); j++) {

            temp = pages.get(j);

            if (indexes.get(temp) < lru) {

                lru = indexes.get(temp);

                pageToBeReplaced = j;

            }

        }

        // Remove the least recently used page

        indexes.remove(pages.get(pageToBeReplaced));

        pages.set(pageToBeReplaced, curPage);

        // Increment page fault count

        page_faults++;

        displayPageFrames(pages, page_faults);

    }
```

```
            // Update the current page index
            indexes.put(curPage, i);
        }
    }
    return page_faults;
}
// Function to display the current state of page frames
static void displayPageFrames(ArrayList<Integer> pages, int page_faults) {
    System.out.print("At PageFault- " + page_faults + " :: Pages- ");
    for (int i = 0; i < pages.size(); i++) {
        System.out.print(" " + pages.get(i));
    }
    System.out.print("\n");
}
// Driver method to test the algorithm
public static void main(String[] args) {
    int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};
    // Number of frames in memory
    pageFrames = 3;
    // Calling LRU page replacement
    System.out.println("--- Implementing Least Recently Used Page Replacement Algorithm -----");
    int pageFaults = lru(pages);
    System.out.println("Number of page faults = " + pageFaults);
}
```

```
}
```