Name :Prasad Lolage

TE B Batch : A Roll no : 3202015

SPOS 4

```
import java.util.*;

public class PageReplacement
{
   static void lru(int[] pages, int frames)
   {
      System.out.println("\nLRU Page Replacement:");
      Set<Integer> memory = new HashSet<>(frames);
      Map<Integer, Integer> indexes = new HashMap<>();
      int pageFaults = 0;

      for (int i = 0; i < pages.length; i++)
      {
         int page = pages[i];
         if (!memory.contains(page))
         {
            if (memory.size() == frames)
            {
               int lruPage = Integer.MAX_VALUE;
               int minIndex = Integer.MAX_VALUE;
               for (int p : memory)
               {
                  int lastUsed = indexes.get(p);
                  if (lastUsed < minIndex)
                  {
                     minIndex = lastUsed;
                     lruPage = p;
                  }
               }
               memory.remove(lruPage);
            }
            memory.add(page);
            pageFaults++;
            System.out.println("Page " + page + " caused a fault.");
         }
         indexes.put(page, i);
      }
      System.out.println("Total Page Faults = " + pageFaults);
   }

   static void optimal(int[] pages, int frames)
   {
      System.out.println("\nOptimal Page Replacement:");
      Set<Integer> memory = new HashSet<>(frames);
      int pageFaults = 0;

      for (int i = 0; i < pages.length; i++)
      {
            int page = pages[i];
            if (!memory.contains(page))
            {
                if (memory.size() == frames)
                {
                    int farthest = i + 1;
                    int pageToRemove = -1;
                    for (int p : memory)
                    {
                        int j;
                        for (j = i + 1; j < pages.length; j++)
                        {
                            if (pages[j] == p) break;
                        }
                        if (j == pages.length)
                        {
                            pageToRemove = p;
                            break;
                        }
                        if (j > farthest)
                        {
                            farthest = j;
                            pageToRemove = p;
                        }
                    }
                    if (pageToRemove == -1)
                    {
                        Iterator<Integer> it = memory.iterator();
                        pageToRemove = it.next();
                    }
                    memory.remove(pageToRemove);
                }
                memory.add(page);
                pageFaults++;
                System.out.println("Page " + page + " caused a fault.");
            }
        }
        System.out.println("Total Page Faults = " + pageFaults);
    }

    public static void main(String[] args)
    {
        int[] pages = {1, 2, 3, 4, 2, 1, 5, 2, 1, 6, 7, 8, 1};
        int frames = 3;
        System.out.println("Page reference string: " + Arrays.toString(pages));
        System.out.println("Number of frames: " + frames);

        lru(pages, frames);
        optimal(pages, frames);
    }
}
```

Output:-

```
Page reference string: [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2]
Number of frames: 4

LRU Page Replacement:
Page 7 caused a fault.
Page 0 caused a fault.
Page 1 caused a fault.
Page 2 caused a fault.
Page 3 caused a fault.
Page 0 caused a fault.
Page 4 caused a fault.
Page 2 caused a fault.
Page 3 caused a fault.
Total Page Faults = 9

Optimal Page Replacement:
Page 7 caused a fault.
Page 0 caused a fault.
Page 1 caused a fault.
Page 2 caused a fault.
Page 3 caused a fault.
Page 4 caused a fault.
Total Page Faults = 6
```