```java
package Assign_2;

import BasicIO.*;

/** This class creates a BasicForm-driven application that keeps print jobs in a linked structure.
  *
  * @author S. Fenwick
  *
  * student # 6005011
  *
  * @version 1.0 (Feb. 2017)                                                     */

public class Main{

  private Node que; //list of print jobs
  private ASCIIDisplayer display;
  public BasicForm form;
  private int jobsLeft;

  /* This constructor creates the BasicForm form and ASCIIDisplayer display, and waits for user interaction. */

  public Main(){

    int button; //button pressed

    display = new ASCIIDisplayer();
    form = new BasicForm("Add Job" , "Print Next" , "Check" , "Quit");
    que = null;

    setUpForm();

    for ( ; ; ) {
      form.clearAll();
      button = form.accept();
      if ( button == 3 ) break;  // Quit
      switch ( button ) {
        case 0: {          // AddJob
          addJob(jobsLeft);
          break;
        }
        case 1: {          // PrintNext
          printNext(jobsLeft);
          break;
        }
        case 2: {          // Check
          check();
          break;
        }
      };
      form.accept("OK");
    };

    form.close();
    display.close();

  } //constructor

  /* This method checks the priority of the current job being added, and if it is a priority 0 sends the job through
   * the addStudent() method (sorted insertion), if it is priority 1 sends the job through the addStaff()
   *  method (sorted insertion), and if it is priority 2 it is  sent through the addFaculty() method
   * (insertion at end of list). It updates the counter whenever a job is added.
   *
   * @param counter keeps count of the jobs being added or removed from the spooler */

  private void addJob(int counter){

    int priority; //student, staff, faculty
    Job aJob;     //job being added

    priority = form.readInt("priority");
    aJob = new Job(form);

    if(priority == 2){
      addFaculty(aJob);
    }
    if(priority == 1){
      addStaff(aJob);
    }
    if(priority == 0){
      addStudent(aJob);
    }

    jobsLeft = counter + 1;

  } //addJob

  /* This method adds a job to the back of the list. Since it is a Faculty job (priority 2) and nothing has a larger
   * priority it does not require a sorted search, and can just be added to the very back of the entire list.
   *
   * @param aJob the current job (node) being added to the spooler (list). */

  private void addFaculty(Job aJob){

    Node  p;
    Node  q;

    q = null;
    p = que;

    while ( p != null ) {
      q = p;
      p = p.next;
    };
    if ( q == null ) {
      que = new Node(aJob,null);
    }
    else {
      q.next = new Node(aJob,null);
    };

    form.writeString("status", "Job Added.");

  } //addFaculty

  /* This method adds a node of priority 1 (staff) to the list. It sorts through the list checking the priority of
   * the other nodes in the list and adds it to the back of the "staff section", i.e it becomes the first node before
   * any node with a priority of 2.
   *
   * @param aJob the current job (node) being added to the spooler (list). */

  private void addStaff(Job aJob){

    Node p;
    Node q;

    q = null;
    p = que;

    while ( p != null && p.item.getPriority() <= aJob.getPriority()){
      q = p;
      p = p.next;
    };

    if(q == null){
      que = new Node(aJob, p);
    }
    else{
      q.next = new Node(aJob, p);

    }

    form.writeString("status", "Job Added.");

  } //addStaff

  /* This method adds a node of priority 0 (student) to the list. It sorts through the list checking the priority of
   * the other nodes in the list and adds it to the back of the "student section", i.e it becomes the first node before
   * any node with a priority of 1.
   *
   * @param aJob the current job (node) being added to the spooler (list). */

  private void addStudent(Job aJob){

    Node p;
    Node q;

    q = null;
    p = que;

    while ( p != null && p.item.getPriority() <= aJob.getPriority()){
      q = p;
      p = p.next;
    };

    if(q == null){
      que = new Node(aJob, p);
    }
    else{
      q.next = new Node(aJob, p);

    }

    form.writeString("status", "Job Added.");

  } //addStudent


  /* This method deletes the first node of the list.
   *
   * @param counter keeps count of the jobs being added or removed from the spooler */

  private void printNext(int counter){

    Job item;

    if(que == null){
      form.writeString("status", "There are no print jobs remaining.");
      jobsLeft = 0;
    }
    else{
      item = que.item;
      que = que.next;
      jobsLeft = counter - 1;
      form.writeString("status", "Job Printed. " + jobsLeft + " Job(s) Remain.");
      writeToDisplay(item);

    }

  } //printNext

  /* This method writes to the ASCIIDisplayer the information of the particular job being printed.
   *
   * @param aJob the current job (node) being printed (deleted). */

  private void writeToDisplay(Job aJob){

    String desc;
    String name;
    int pages;

    desc = aJob.getDescription();
    name = aJob.getName();
    pages = aJob.getPages();

    display.writeString("(" + name + " prints " + desc + ": " + pages + " pages.)");
    display.newLine();

  } //writeToDisplay

  /* This method checks how many pages are remaining in the spooler (que). */

  private void check(){

    Node p;
    int pageCount = 0;

    p = que;

    while( p != null){
      pageCount = pageCount + p.item.getPages();
      p = p.next;
    }

    form.writeString("status", "There are " + pageCount + " pages in the que.");

  } //check

  /* This method sets up the BasicForm (form). */

  private void setUpForm(){

    form.setTitle("Print Spooler");

    form.addTextField("description","Description:", 15, 10, 10);

    form.addTextField("send","Sender:", 10, 10, 40);

    form.addTextField("pages","# Pages:", 6, 10, 70);

    form.addRadioButtons("priority","Priority",true, 248, 8, Job.PRIORITY);

    form.addTextField("status","Status:", 39, 10, 100);
    form.setEditable("status", false);

  } //setUpForm

  public static void main (String args[]) {Main m = new Main();}

} //Main
```