```
import java.io.File;

import java.io.FileWriter;

import java.io.FileNotFoundException;

import java.util.InputMismatchException;

import java.io.*;

import java.util.Scanner;

import java.util.ArrayList;


public class LibraryApplication {

   private static Item[] item = new Item[20];//Array of 20 objects from Item class

   private static User[] user = new User[10];//Array of 10 objects from User class

   private static ArrayList<Loan> loans = new ArrayList<Loan>();//Array list using Loan class to create objects. Allows for additional objects to be added later.

   private static String[] headers; //Stores headings to be used when writing back to file and used in display as well.

   int positionLoan, positionItem, positionUser; //Used to find ArrayList position of loan, item and user


   public static void main(String[] args) {

      LibraryApplication app = new LibraryApplication();

      app.processItems(); // Calls on three processing methods

      app.processUsers();

      app.processLoans();

      app.menu(); // Calls the main menu

   }


      private void menu(){ //Main menu

      int option = 0;

      while (option != 6){ //Loops menu so it appears after each selection unless exit option has been choosen

         System.out.println("-----------------Library Application-----------------\n"); //Displays menu

         System.out.println("Create a loan(1)\t\tRenew an exisiting loan(2)\n");
```

```
System.out.println("Process a return(3)\t\tView all items(4)\n");
System.out.println("View all loans(5)\t\t\tExit(6)\n");
System.out.println("-----------------------------------------------------");
System.out.println("What would you like to do? (Options 1-6)");
try{ //Catches inputs that aren't an integer number
   Scanner myScanner = new Scanner(System.in);
   option = myScanner.nextInt(); // myScanner picks up on what option has been selected
   switch(option){
      case 1: //Option 1
         this.createLoan();
         break;
      case 2: //Option 2
         this.renew();
         break;
      case 3: //Option 3
         this.returnItem();
         break;
      case 4: //Option 4
         this.viewItems();
         break;
      case 5: //Option 5
         this.viewLoans();
         break;
      case 6: // Option 6
         System.out.println("\nSaving to file...");
         System.out.println("\n\nThank you for using this library application!\nGoodbye!");
         this.printToFile();
         break;
      default: //Invalid option check
         System.out.println("Please enter a valid selection");
   }
```

```
        }
        catch (InputMismatchException e){
            System.out.println("Please enter a valid selection!");
        }
    }
}

  private void processItems(){
    int x = 0;
    try {
    File file = new File("ITEMS(1).csv");
    Scanner scanner = new Scanner(file);
    String data = scanner.nextLine(); //Skips first line - titles
    while (scanner.hasNext()) {
        data = scanner.nextLine(); // Reads in whole line into data variable
        String [] dataArray = data.split(","); // Splits data into array using comma
        item[x] = new Item(dataArray[0], dataArray[1], dataArray[2], //Creates new object using data
array
              dataArray[3], dataArray[4], dataArray[5]);
        x++;
    }
    scanner.close();
    }
    catch (FileNotFoundException e) { //Catches error if file isn't found
    System.out.println("Error - Item file not found");
    }
  }
  private void processUsers(){
    int x = 0;
    try {
    File file = new File("USERS(1).csv");
```

```
        Scanner scanner = new Scanner(file);
        String data = scanner.nextLine();
        while (scanner.hasNext()) {
          data = scanner.nextLine(); // Reads in whole line into data variable
          String [] dataArray = data.split(","); // Splits data into array using comma
          user[x] = new User(dataArray[0], dataArray[1], dataArray[2], //Creates new object using data
array
                dataArray[3]);
          x++;
        }
        scanner.close();
        }
        catch (FileNotFoundException e) { //Catches error if file isn't found
          System.out.println("Error - User file not found");
        }
    }

    private void processLoans(){
        try {
          File file = new File("LOANS(1).csv");
          Scanner scanner = new Scanner(file);
          String data = scanner.nextLine();
          headers = data.split(",");
          while (scanner.hasNext()){
            data = scanner.nextLine();
            String [] dataArray = data.split(",");
            for(int x = 0; x < item.length; x++){
              if (item[x].getItemCode().equals(dataArray[0])){ //Checks through all items to see if
barcode exists and records position
                positionItem = x;
              }
            }
```

```
            if(item[positionItem].getType().equals("Book")){
               loans.add(new Book(dataArray[0], dataArray[1], dataArray[2],
            dataArray[3], dataArray[4]));
            }
            else{
               loans.add(new Multimedia(dataArray[0], dataArray[1], dataArray[2],
            dataArray[3], dataArray[4]));
            }
         }
         scanner.close();
      }
      catch (FileNotFoundException e) {
         System.out.println("Error - Loan file not found");
      }
   }

   private void createLoan(){
      String search; //To store users search
      positionUser = 0; //To hold location of user in user array
      positionItem = 0; //To hold location of item in item array
      boolean foundUser = false;
      boolean foundItem = false;
      Scanner scanner = new Scanner(System.in);
      System.out.println("-----------------Create a loan-----------------");
      System.out.println("Please enter the user ID: ");
      search = scanner.next();
      for(int x = 0; x < user.length && foundUser == false; x++){
         if (user[x].getUserId().equals(search)){ //Checks through all users to see if id exists and
records position
            positionUser = x;
            foundUser = true;
```

```
        }
    } //If user exists, moves onto item query
    if (foundUser == true){
        System.out.println("Please enter the item ID: ");
        search = scanner.next();
        for(int y = 0; y < item.length && foundItem == false; y++){
            if (item[y].getItemCode().equals(search)){ //Checks through all items to see if barcode exists and records position
                positionItem = y;
                foundItem = true;
            }
        } //If item exists too, goes through process of creating new loan object
        if (foundItem == true){
            System.out.println("Adding item loan...");
            if(item[positionItem].getType().equals("Book")){
                loans.add(new Book(item[positionItem].getItemCode(), user[positionUser].getUserId())); //If id and barcode exists, creates object adding onto arrayList
                loans.get(loans.size() - 1).setDueDate();
            }
            else{
                loans.add(new Multimedia(item[positionItem].getItemCode(), user[positionUser].getUserId())); //If id and barcode exists, creates object adding onto arrayList
                loans.get(loans.size() - 1).setDueDate();
            }
            System.out.println(headers[0] + "\t\t" + headers[1] + "\t\t" + headers[2] + "\t" + headers[3] + "\t" + headers[4]);
            loans.get(loans.size() - 1).displayLoan(); //Prints out new loan
        }
        else
            System.out.println("No item found with that ID");
        }
    else
```

```
            System.out.println("No user found with that ID");
        }

        private void viewItems(){ //Displays all item objects in program
            System.out.println("-----------------Item Report-----------------");
            System.out.printf( "%-15s%-22s%-42s%-22s%-22s%-12s\n" ,"Barcode"
,"Author","Title","Type","Year","ISBN");//Formatted headings
            for(int x = 0; x < item.length; x++){
                item[x].displayItem(); // Goes through array of objects and calls display method for each one
            }
        }

        private void viewLoans(){
            System.out.println("-----------------Loan Report-----------------");
            System.out.println(headers[0] + "\t\t" + headers[1] + "\t\t" + headers[2] + "\t" + headers[3] +
"\t" + headers[4]);
            for(int x = 0; x < loans.size(); x++){
                loans.get(x).displayLoan();
            }
        }

        private void returnItem(){ //Used to process a return of an item on loan
            int positionLoan = 0; //Used to find ArrayList position of loan

            Boolean found = false;
            String search; // Holds users barcode search
            System.out.println("-----------------Return-----------------");
            System.out.println("What loan would you like to return? ");
            System.out.println("Please enter the barcode of the loan you would like to return: ");
            Scanner scanner = new Scanner(System.in);
            search = scanner.next(); //User inputs the loan they are searching for
            for(int x = 0; x < loans.size() && found == false; x++){
```

```
        if (search.equals(loans.get(x).getLoanCode())){ //Itterates through loans until finds a match or comes to the end
            System.out.println("Match found");
            positionLoan = x; //Records position in loan list of where it was found
            found = true;
        }
    }
    if (found == true)
    {
        if (loans.get(positionLoan).checkDate()==true){
            loans.remove(positionLoan);
            System.out.println("Loan returned.");
        }
        else{
            loans.remove(positionLoan);
            System.out.println("Loan was returned past due date. Consult overdue policy!"); //As no overdue loan process has been specified, outputs error message
        }
    }
    else
        System.out.println("No barcode found");
  }

  private void renew(){ //Method to renew current loans - option 2
    positionLoan = 0;
    positionItem = 0;
    Boolean found = false;
    String search; // Holds users barcode search
    System.out.println("-----------------Renew-----------------");
    System.out.println("What loan would you like to renew? ");
    System.out.println("Please enter the barcode you would like to search for: ");
    Scanner scanner = new Scanner(System.in);
```

```
        search = scanner.next(); //User inputs the loan they are searching for
        for(int x = 0; x < loans.size() && found == false; x++){
            if (search.equals(loans.get(x).getLoanCode())){ //Itterates through loans until finds a match or comes to the end
                System.out.println("-----------------Match Found-----------------");
                System.out.println(headers[0] + "\t\t" + headers[1] + "\t\t" + headers[2] + "\t" + headers[3] + "\t" + headers[4]);
                loans.get(x).displayLoan();
                System.out.println("-----------------------------------------------");
                positionLoan = x; //Records position in loan list of where it was found
                found = true;
            }
        }
        if (found == true){ //If loan exists, calls checkRenews function
            loans.get(positionLoan).checkRenews();
        }
        else
            System.out.println("No match found"); //Prints if no matching loan barcode is found
    }

    private void printToFile(){ //Prints all loan objects back into LOANS(1).csv one line at a time using a comma as a cell seperator
        try{
            FileWriter writer = new FileWriter("LOANS(1).csv", false);
            for (int x = 0; x < headers.length; x++){
                writer.write(headers[x]);
                writer.write(",");
            }
            writer.write("\n");
            for (int y = 0; y < loans.size(); y++){
                writer.write(loans.get(y).toPrint());
                writer.write("\n");
```

```
            }
            writer.close();
        }
        catch(IOException e){
            System.out.println("Error trying to save. Please ensure LOANS(1).csv is not open during
application operation to ensure data integrity!");
        }
    }
}

public class Item {
    private String barcode, author, title, type, year, isbn;

    //Constructor to create new item instance.
    public Item(String barcode, String author, String title,
            String type, String year, String isbn){
        this.barcode = barcode;
        this.author = author;
        this.title = title;
        this.type = type;
        this.year = year;
        this.isbn = isbn;
    }

    //Displays item in formatted collumns
    public void displayItem(){//https://stackoverflow.com/questions/39312589/aligning-columns
        System.out.printf("%-15s%-22s%-42s%-22s%-22s%-12s\n",barcode,author,title,type,year ,isbn);
//Prints formated objects
    }

    public String getItemCode(){ //Getter for barcode
        return barcode;
```

```
    }

    public String getType(){ //Getter for item type
        return type;
    }
}

import java.time.format.DateTimeFormatter;
import java.time.LocalDate;

public abstract class Loan{
    DateTimeFormatter format = DateTimeFormatter.ofPattern("dd/MM/yyyy");
    protected LocalDate currentDate;
    private LocalDate issueDate, dueDate;
    private int renews = 0;
    private String barcode, userId;

    public Loan(String barcode, String userId, String issueDate, String dueDate,
            String renews){ //Constructor for loans being read in from .csv file
        this.barcode = barcode;
        this.userId = userId;//Calls on parent class constructor
        this.issueDate=LocalDate.parse(issueDate, format);
        this.dueDate=LocalDate.parse(dueDate, format);
        this.renews=Integer.parseInt(renews);
    }

    public Loan(String barcode, String userId){//Overloaded constructor allows for new loans to be
processed differently.
        this.barcode = barcode;
        this.userId = userId;
        this.issueDate = issueDate.now();
```

```
        this.renews = 0;
    }

    public void setDueDate(){ //Overridden
        this.dueDate = null;
    }

    public void displayLoan(){ //Overridden
        System.out.println(barcode + "\t" + userId + "\t" + format.format(issueDate) + "\t" +
format.format(dueDate) + "\t" + renews);
    }

    public int getRenews(){ //Getter for renews
        return renews;
    }

    public String getLoanCode(){ //Getter for barcode
        return barcode;
    }

    public String getUserId(){ //Getter for user ID
        return userId;
    }

    public LocalDate getDueDate(){ //Getter for due Date
        return dueDate;
    }

    public void updateDueDate(){ //Overridden
        this.dueDate = null;
    }
```

```
    public void updateRenews(){ //Overridden
        renews++;
    }

    public boolean checkDate(){ //Overridden
        if (currentDate.now().isBefore(dueDate)|| currentDate.now().isEqual(dueDate)){
            return true;
        }
        else return false;
    }

    public String toPrint(){ //Overridden
        return barcode + "," + userId+ "," + format.format(issueDate) + "," + format.format(dueDate) +
"," + renews;
    }

    public LocalDate getIssueDate(){ //Getter for Issue Date
        return issueDate;
    }

    public void checkRenews(){ //Overridden
    }
}

public class User {
    private String userId, firstName, lastName, email;

    //Constructor to create new instance of User, read in from csv
    public User(String userId, String firstName, String lastName, String email){
        this.userId = userId;
```

```
        this.firstName= firstName;
        this.lastName = lastName;
        this.email = email;
    }

    public String getUserId(){ //Getter for User Id
        return userId;
    }
}

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class Multimedia extends Loan{ //Child of Loan class
    DateTimeFormatter format = DateTimeFormatter.ofPattern("dd/MM/yyyy");
    private LocalDate dueDate;
    private int MAX_RENEWS = 2; //Used for maxiumum renews

    //Constructor to take in loans being read in from csv
    public Multimedia(String barcode, String userId, String issueDate, String dueDate, String renews){
        super (barcode, userId, issueDate, dueDate, renews); //Calls parent constructor
        this.setDueDate(); //Assigns due date as this is item type specific
    }

    //Constructor to take in loans being created during program operation
    public Multimedia(String barcode, String userId){//Overloaded constructor allows for new loans to be processed differently.
        super(barcode, userId);
    }

    public void updateDueDate(){ //Updates due date by one week
```

```
        this.dueDate = dueDate.plusWeeks(1);
    }

    public void setDueDate(){ //Sets due date by one week
        this.dueDate = this.getIssueDate().plusWeeks(1);
    }

    public void displayLoan(){ //Overrides parent and displays loan
        System.out.println(this.getLoanCode() + "\t" + this.getUserId() + "\t" +
format.format(this.getIssueDate()) + "\t" + format.format(dueDate) + "\t" + this.getRenews());
    }

    public String toPrint(){ //Overides parent and returns string that can be written to file
        return this.getLoanCode() + "," + this.getUserId()+ "," + format.format(this.getIssueDate()) + ","
+ format.format(dueDate) + "," + this.getRenews();
    }

    public LocalDate getDueDate(){ //Getter for due Date
        return dueDate;
    }

    public boolean checkDate(){ //Overrides parent, checks if date is before or on due date
        if (currentDate.now().isBefore(dueDate)|| currentDate.now().isEqual(dueDate)){
            return true;
        }
        else return false;
    }

    public void checkRenews(){ //Checks current renews again Max renews variable
        if(this.getRenews()>= MAX_RENEWS){ //Outputs error message
            System.out.println("Too many renews! Multimedia can have no more than 2 renews. Please
return by due date: " + format.format(dueDate));
```

```
        }
        else{ //Starts process of renewing loan
            System.out.println("Renewing");
            this.updateRenews();
            this.updateDueDate();
            System.out.println("New due date: " + format.format(dueDate)); //Prints new due date for
loan
        }
    }
}

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class Book extends Loan{ //Child of Loan class
    DateTimeFormatter format = DateTimeFormatter.ofPattern("dd/MM/yyyy");
    private LocalDate dueDate;
    private int MAX_RENEWS = 3;

    //Constructor for intake of loans being read in from
    public Book(String barcode, String userId, String issueDate, String dueDate, String renews){
        super (barcode, userId, issueDate, dueDate, renews); //Calls parent class constructor
        this.setDueDate();
    }

    public Book(String barcode, String userId){//Overloaded constructor allows for new loans to be
processed differently.
        super(barcode, userId);
    }

    public void updateDueDate(){ //Updates due date by two weeks
        this.dueDate = dueDate.plusWeeks(2);
```

```
    }

    public void setDueDate(){ //Sets due date as four weeks from current date
        this.dueDate = this.getIssueDate().plusWeeks(4);
    }

    public void displayLoan(){ //Overrides parent method, displays loan
        System.out.println(this.getLoanCode() + "\t" + this.getUserId() + "\t" +
format.format(this.getIssueDate()) + "\t" + format.format(dueDate) + "\t" + this.getRenews());
    }

    public String toPrint(){ //Overrides parent, returns printable string for writing to file
        return this.getLoanCode() + "," + this.getUserId()+ "," + format.format(this.getIssueDate()) + ","
+ format.format(dueDate) + "," + this.getRenews();
    }

    public LocalDate getDueDate(){ //Getter for due Date
        return dueDate;
    }

    public boolean checkDate(){ //Checks if date is before or on due date
        if (currentDate.now().isBefore(dueDate)|| currentDate.now().isEqual(dueDate)){
            return true;
        }
        else return false;
    }

    public void checkRenews(){ //Checks current renews against max
        if(this.getRenews()>= MAX_RENEWS){ //Outputs error if too many
            System.out.println("Too many renews! Books can have no more than 3 renews. Please return
by due date: " + format.format(dueDate));
        }
```

```
        else{ //Renews if appropriate
            System.out.println("Renewing");
            this.updateRenews();
            this.updateDueDate();
            System.out.println("New due date: " + format.format(dueDate));
        }
    }
}
```