BorrowDlg:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

/**
 * BorrowDlg is a custom class to add functionality to issue items on loan to a borrower
 * BorrowDlg has been created to fulfill the needs of the specification
 *
 * @author Andrew Robson W16011147
 * @version 1.0
 */
public class BorrowDlg extends JDialog implements ActionListener
{
   private MainMenu parent;
   private JTextField borrowerField, itemField;
   private JButton submitButton, cancelButton;

   /**
    * This method is primarily used to create the GUI
    * creates the panels
    * creates the buttons
    * creates the text fields
    * adds action listeners to the buttons
    */
   public BorrowDlg(MainMenu p)
   {
      setTitle("Register Loan of an Item");
```

```
        parent = p; //Parent is MainMenu

        borrowerField = new JTextField(10);//sets the borrowerField attribute
        itemField = new JTextField(10);//sets the itemField attribute
        submitButton = new JButton("Submit");//sets the submit button
        cancelButton = new JButton("Cancel");//sets the cancelButton

        //Creates the panel and adds all the text fields and labels
        JPanel thePanel = new JPanel();
        thePanel.add(new JLabel("Borrower ID:"));
        thePanel.add(borrowerField);
        thePanel.add(new JLabel("Item ID:"));
        thePanel.add(itemField);
        add(thePanel, BorderLayout.CENTER);//Will display in the center of the panel

        //Creates the panel and adds the buttons to it
        thePanel = new JPanel();
        thePanel.add(submitButton);
        thePanel.add(cancelButton);
        add(thePanel, BorderLayout.SOUTH);//Will display south of the panel

        setBounds(200, 200, 800, 200);//Sets the dimensions

        submitButton.addActionListener(this);//adds an actionListener to the submit button
        cancelButton.addActionListener(this);//adds an actionListener to the cancel button
    }

    /**
     * Assigns all of the fnctionality to the buttons when they are pressed
```

```
     */
    public void actionPerformed(ActionEvent evt)
    {
        Object src = evt.getSource();//sets the src variable
        if (src == submitButton)
        {
            processLoan();//calls the processLoan function
            borrowerField.setText("");
            itemField.setText("");
        }
        else if (src == cancelButton)
        {
            setVisible(false);//Hides the panel
            borrowerField.setText("");
            itemField.setText("");
        }
    }

    /**
     * Implements all of the functionality that has been declared in the specification
     * Particularly the requirements relevent to part 1
     * Displays errors to the user in the panel
     */
    public void processLoan()
    {
        try
        {
            Integer borrowerID = new Integer(borrowerField.getText());
            Borrower borrower = parent.getBorrowers().get(borrowerID);
```

```
        Integer itemID = new Integer (itemField.getText());
        Item item = parent.getItems().get(itemID);
        if (borrower == null)//If borrowe is null then do the following
        {
           //Displays an error
           JOptionPane.showMessageDialog(this, "Borrower could not be found",
              " Error: ", JOptionPane.ERROR_MESSAGE);
           return;
        }

        if (item == null)//If item is null then do the following
        {
           //Displays an error
           JOptionPane.showMessageDialog(this, "Item could not be found",
              " Error: ", JOptionPane.ERROR_MESSAGE);
           return;
        }

        Borrower isBorrowed = item.getBorrowedBy();

        if (isBorrowed != null)//If isBorrowed is not null then do the following
        {
           //Displays an error
           JOptionPane.showMessageDialog(this, "Item is on loan",
              " Error: ", JOptionPane.ERROR_MESSAGE);
           return;
        }

        for(LoanTransaction loan: parent.getLoans()) {//for every loan do the following
```

```
            if (borrower.equals(loan.getBorrower()))//If these two values are equal then do the following
            {
               if((System.currentTimeMillis() - loan.getTimeStamp()) > parent.getLOANMAX())
               //Takes the time in milliseconds, removes the timeStamp variable and compares it to
LOANMAX
               //which is in the MainMenu class
               {
                  //Displays an error
                  JOptionPane.showMessageDialog(this, "One or more loans have been discovered as
overdue",
                     " Error: ", JOptionPane.ERROR_MESSAGE);
                  return;
               }
            }
         }

         //adds a new loan transaction
         parent.getLoans().add(new LoanTransaction(borrower, item, System.currentTimeMillis()));

         item.setBorrowedBy(borrower);//links the item with a borrower
         System.out.println("Loan has been successfully created");//Prints to console
      }
      catch (NumberFormatException ex)
      {
         //Displays error
         JOptionPane.showMessageDialog(this, ex.getMessage(), "Number format error",
JOptionPane.ERROR_MESSAGE);
      }
   }
}
```

Main Menu:

```
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.util.*; //ArrayList; HashMap; LinkedList;

/**
 * Graphical user interface
 * Also contains Map of Integers (borrower IDs) to Borrowers,
 *            Map of Integers (item IDs) to Items
 *            List of Loan records
 */
public class MainMenu extends JFrame implements ActionListener {
  public static final long LOANMAX = 1814400000; //ms = 21 days
  // data collections
  private Map<Integer, Borrower> borrowers;
  private Map<Integer, Item> items;
  private List<LoanTransaction> loans;

  //GUI
  private ReturnDlg returnDlg;
  private BorrowDlg borrowDlg;
  private JButton btnReadData, btnSaveLoans, btnLendItems, btnReturnItems,
            btnListLoans, btnListODLoans;

  public static void main(String[] args) {
    MainMenu app = new MainMenu();
```

```
   app.setVisible(true);
  }

  /**
   * Initialise the data stores and main menu
   *   Menu consists of JButtons with current class as ActionListener
   */
  public MainMenu() { //constructor
   // Database
   borrowers = new HashMap<Integer, Borrower>();
   items = new HashMap<Integer, Item>();
   loans = new LinkedList<LoanTransaction>();

   // GUI - create custom dialog instances
   returnDlg = new ReturnDlg(this);
   borrowDlg = new BorrowDlg(this);

   // GUI - set window properties
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   setBounds(400, 200, 500, 600);

   //GUI - main menu buttons
   JPanel mainPnl = new JPanel();
   mainPnl.setLayout(new GridLayout(3,2));

   btnReadData = new JButton("Read Data");
   btnReadData.addActionListener(this);
   mainPnl.add(btnReadData);
```

```
    btnLendItems = new JButton("Lend Items");
    btnLendItems.addActionListener(this);
    mainPnl.add(btnLendItems);

    btnReturnItems = new JButton("Return Items");
    btnReturnItems.addActionListener(this);
    mainPnl.add(btnReturnItems);

    //Buttons for task 2
    btnListLoans = new JButton("List Loans");
    btnListLoans.addActionListener(this);
    mainPnl.add(btnListLoans);

    btnListODLoans = new JButton("List Overdue Loans");
    btnListODLoans.addActionListener(this);
    mainPnl.add(btnListODLoans);

    btnSaveLoans = new JButton("Save Loans");
    btnSaveLoans.addActionListener(this);
    mainPnl.add(btnSaveLoans);

    add(mainPnl, BorderLayout.CENTER);
  } //end constructor

  /**
   * Accessors for data structures
   */
  public Map<Integer, Borrower>  getBorrowers() { return borrowers; }
  public Map<Integer, Item>      getItems()     { return items; }
```

```
 public List<LoanTransaction> getLoans()       { return loans; }
 public long getLOANMAX() { return LOANMAX; } //Created as a variable to be used within other classes


 /**
  * Actions in response to buttons
  */
 public void actionPerformed(ActionEvent evt) {
  Object src = evt.getSource();
  //read borrowers, items, loans JUST ONCE to initialise the system
  if (src == btnReadData) {
    readBorrowerData();
    listBorrowers();
    readItemData();
    readLoans(); // saved from a previous session
    listItems(); // AFTER loans reloaded
    btnReadData.setEnabled(false);
  } else if (src == btnLendItems) { // borrowDlg dialog will do multiple loans
    borrowDlg.setVisible(true);
  } else if (src == btnReturnItems) { // returnDlg will do multiple returns
    returnDlg.setVisible(true);
  } else if (src == btnListLoans) {//ListLoans will list multiple loans
    listLoans();
  }
  else if (src == btnListODLoans) {//ListODLoans will list all overdue loans
    listODLoans();
  }
  else if (src == btnSaveLoans)//SaveLoans saves loans to a text file
  {
    saveLoans();
```

```
  }
 }

 /**
  * Read data from borrowers.txt using a Scanner; unpack and populate
  *   borrowers Map. List displyed on console.
  */
 public void readBorrowerData() {
  String fnm="", snm="", pcd="";
  int num=0, id=1;
  try {
    Scanner scnr = new Scanner(new File("borrowers.txt"));
    scnr.useDelimiter("\\s*#\\s*");
    while (scnr.hasNextInt()) {
      id  = scnr.nextInt();
      snm = scnr.next();
      fnm = scnr.next();
      num = scnr.nextInt();
      pcd = scnr.next();
      borrowers.put(new Integer(id), new Borrower(id, snm, fnm, num, pcd));
    }
    scnr.close();
  } catch (NoSuchElementException e) {
    System.out.printf("%d %s %s %d %s\n", id, snm, fnm, num, pcd);
    JOptionPane.showMessageDialog(this, e.getMessage(),
        "fetch of next token failed ", JOptionPane.ERROR_MESSAGE);
  } catch (NumberFormatException e) {
      JOptionPane.showMessageDialog(this, e.getMessage(),
          "Error", JOptionPane.ERROR_MESSAGE);
```

```
  } catch (IllegalArgumentException e) {
     JOptionPane.showMessageDialog(this, e.getMessage(),
        "Error", JOptionPane.ERROR_MESSAGE);
  } catch (IOException e) {
    JOptionPane.showMessageDialog(this, "Borrowers file not found",
       "Unable to open data file", JOptionPane.ERROR_MESSAGE);
  }
 } //end readBorrowerData

 /**
  * List Borrowers on console
  */
 public void listBorrowers() {
  System.out.println("Borrowers:");
  for (Borrower b: borrowers.values()) {
    System.out.println(b);
  }
  System.out.println();
 }

 /**
  * Read data from items.txt using a Scanner; unpack and populate
  *   items Map. List displyed on console.
  */
 public void readItemData() {
  String ttl="", aut="";
  int id=1;
  try {
    Scanner scnr = new Scanner(new File("items.txt"));
```

```
    scnr.useDelimiter("\\s*#\\s*");
    while (scnr.hasNextInt()) {
      id  = scnr.nextInt();
      ttl = scnr.next();
      aut = scnr.next();
      items.put(new Integer(id), new Item(id, ttl, aut));
    }
    scnr.close();
   } catch (NoSuchElementException e) {
    System.out.printf("%d %s %s\n", id, ttl, aut);
    JOptionPane.showMessageDialog(this, e.getMessage(),
        "fetch of next token failed ", JOptionPane.ERROR_MESSAGE);
   } catch (NumberFormatException e) {
      JOptionPane.showMessageDialog(this, e.getMessage(),
          "Error", JOptionPane.ERROR_MESSAGE);
   } catch (IllegalArgumentException e) {
      JOptionPane.showMessageDialog(this, e.getMessage(),
          "Error", JOptionPane.ERROR_MESSAGE);
   } catch (IOException e) {
    JOptionPane.showMessageDialog(this, "Items file not found",
        "Unable to open data file", JOptionPane.ERROR_MESSAGE);
   }
 } //end readItemData

 /**
  * List Items on console
  */
 public void listItems() {
   System.out.println("Items:");
```

```
  for (Item i: items.values()) {
    System.out.println(i);
  }
  System.out.println();
 }

 //Assumes borrowers, items have been loaded
 public void readLoans() {
  if (loans.size() > 0) {
    JOptionPane.showMessageDialog(this, "Already some loans!",
        "Error", JOptionPane.ERROR_MESSAGE);
    return;
  }
  try {
    Scanner scnr = new Scanner(new File("loans.txt"));
    while (scnr.hasNextInt()) {
      Borrower b = borrowers.get(scnr.nextInt());
      Item i     = items.get(scnr.nextInt());
      LoanTransaction t = new LoanTransaction(b, i, scnr.nextLong());
      loans.add(t);
      i.setBorrowedBy(b);
    }
    System.out.printf("%d loan records added\n", loans.size());
  } catch (IOException e) {
    JOptionPane.showMessageDialog(this, "Loans file not found",
        "Unable to open data file", JOptionPane.ERROR_MESSAGE);
  }
 } //end readloans
```

```
 /**
  * Lists all the overdue books to the console
  */
 public void listODLoans()
 {
    System.out.println("Overdue Loans:");
    for (LoanTransaction loan: loans)//for every loan do the following
    {
      if((System.currentTimeMillis() - loan.getTimeStamp()) > LOANMAX)
      //Gets the time in milliseconds whilst removing the timestamp from itself, and compares it to
LOANMAX
      {
       System.out.println(loan);//Prints out a loan
      }
    }
    System.out.println();
 }//end listODLoans


 /**
  * Lists all of the loans within the system
  */
 public void listLoans()
 {
    System.out.println("Loans:");
    for (LoanTransaction i: loans)//for every loan do the following
    {
       System.out.println(i);//Prints out loan to console
    }
    System.out.println();
```

```
 }//end listLoans


 /**
  * Saves the loans to a text file (currently loans.txt)
  */
 public void saveLoans()
 {
  try
  {
    PrintWriter writer = new PrintWriter(new File("loans.txt"));//Declares print wwriter
    if (loans.size() > 0)//If loan size greater then 0 continue
    {
       for(LoanTransaction loan: loans)//for every loan do the following
       {
          writer.println(loan.getBorrower().getBwrID() + " " + loan.getItem().getItemID() + " " +
          loan.getTimeStamp());
          //Print out the following data to the text file
       }
    }
    else
    {
       writer.println("");//Writes nothing to the file
    }
    writer.close();//Closes the connection to the file
  }
  catch (IOException e)
  {
     System.out.println("Loans file not found");
     //Prints out an error message
```

```
    }
  }//end saveLoans
} //end class
```

ReturnDlg:

```
/**
 * ReturnDlg
 * Custom dialog class with methods to get details of a loan,
 *  and to cancel it, deleting the loan transaction record
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*; //Date, collection classes

public class ReturnDlg extends JDialog implements ActionListener {
  private MainMenu parent;
  private JTextField txtItemID;
  private JButton btnSubmit, btnHide;

  public ReturnDlg(MainMenu p) {
    setTitle("Register Return of an Item");
    parent = p; //data structures are here

    //Components -
    txtItemID = new JTextField(10); //input field, 10 columns wide
    btnSubmit = new JButton("Submit");
    btnHide   = new JButton("Hide");

    //Layout -
    JPanel pnl = new JPanel();
    pnl.add(new JLabel("Item code:"));
    pnl.add(txtItemID);
```

```
    add(pnl, BorderLayout.CENTER);

    pnl = new JPanel();
    pnl.add(btnSubmit);
    pnl.add(btnHide);
    add(pnl, BorderLayout.SOUTH);

    setBounds(100, 100, 400, 100);



    //Action
    btnSubmit.addActionListener(this);
    btnHide.addActionListener(this);
  } //end constructor

  /**
   * Actions: on click of 'Submit', find loan record and delete from database;
   *          on click of 'Hide', hide the dialogue window.
   */
  public void actionPerformed(ActionEvent evt) {
    Object src = evt.getSource();
    if (src == btnHide) {
      setVisible(false);
      txtItemID.setText("");
    }
    else if (src == btnSubmit) {
      processReturn();
      txtItemID.setText("");
    }
```

```
 } //end actionPerformed



 /**
  * Does the actual work of finding a loan record and deleting it;
  * Also updates 'borrowedBy' field of Item.
  */
 public void processReturn() {
  try {
    Integer itmID = new Integer(txtItemID.getText());
    Item itm = parent.getItems().get(itmID);
    if (itm == null) {
    JOptionPane.showMessageDialog(this, "Item not found",
         "Error", JOptionPane.ERROR_MESSAGE);
     return;
    }
    Borrower bby = itm.getBorrowedBy();
    if (bby == null) {
     JOptionPane.showMessageDialog(this, "Not on loan",
         "Error", JOptionPane.ERROR_MESSAGE);
     return;
    }
    //Find LoanTransaction ...
    LoanTransaction mrkr = null;
    for (LoanTransaction t: parent.getLoans()) {
     if (t.getBorrower() == bby && t.getItem() == itm) {
       mrkr = t;
       break;
     }
```

```
      }
      if (mrkr == null) {
        JOptionPane.showMessageDialog(this, "Loan transaction not found",
            "Error", JOptionPane.ERROR_MESSAGE);
        return;
      }
      //Otherwise ...
      parent.getLoans().remove(mrkr);
      itm.clearBorrowedBy();
      System.out.printf("Item returned: [%s]\n\n", itm);
    } catch (NumberFormatException ex) {
      JOptionPane.showMessageDialog(this, ex.getMessage(),
          "Number format error", JOptionPane.ERROR_MESSAGE);
    }
  }
}
```

LoanTransaction:

```
import java.util.Date;

/**
 * LoanTransaction
 * class to record borrowing of items
 */
public class LoanTransaction {
  private Borrower bdBy;
  private Item item;
  private long timeStamp;

  /**
   * Constructor of Loan record with given Borrower, Item,
   *   long integer timestamp
   */
  public LoanTransaction(Borrower b, Item i, long ts) {
    bdBy = b;
    item = i;
    timeStamp = ts;
  }

  public Borrower getBorrower() { return bdBy; }
  public Item getItem()        { return item; }
  public long getTimeStamp()    { return timeStamp; }

  public String toString() {
    return String.format("Item [%s] borrowed by [%s]; timestamp %d",
                         item, bdBy, timeStamp);
```

```
  }

  /**
   * Make a string representation of the Lone without the borrower reference
   *
   * In most contexts, Item has correct Borrower reference
   */
  public String toStringOmitBwr() {
    return String.format("Item [%s] timestamp %d", item, timeStamp);
  }
}
```

Borrower:

```
/**
 * Borrower
 * Encapsulates a registered user of the library
 */
public class Borrower {
  private int bwrID;
  private String surName;
  private String frstName;
  private int houseNumber;
  private String postCode;

  /**
   * Construct a Borrower with given ID, surname, forename,
   *   house number, postcode
   */
  public Borrower(Integer bID, String sn, String fn, int hn, String pcd)
     throws IllegalArgumentException {
   if (!dataValid(bID, sn, fn, hn, pcd)) {
     throw new IllegalArgumentException("Bad arg(s) in Borrower constructor");
   }
   bwrID = bID;
   surName = sn;
   frstName = fn;
   houseNumber = hn;
   postCode = pcd;
  }

  public Integer getBwrID()   {  return bwrID;      }
```

```
  public String getSurName()  {  return surName;     }
  public String getFrstName() {  return frstName;    }
  public int getHouseNumber() {  return houseNumber; }
  public String getPostCode() {  return postCode;    }

  public String toString() {
    return String.format("%03d: %s %s (%d, %s)",
      bwrID, frstName, surName, houseNumber, postCode);
  }

  /**
   * return true if data are valid for forming a Borrower:
   *   surname, forename are strings of 1 or more characters,
   *   ID, house number are not negative,
   *   postcode conforms to simplified regular expresson
   */
  public static boolean dataValid(
            int id, String snm, String fnm, int num, String pcd) {
    if (snm.length() == 0) return false;
    if (fnm.length() == 0) return false;
    if (id < 0 || num < 0) return false;
    String pcdRegex = "[A-Z]{2}[0-9]{1,2}\\s[0-9][A-Z]{2}";
      //"most" but not all valid postcodes (Google!)
    if (!pcd.matches(pcdRegex)) return false;
    return true;
  }
}
```

Item:

```
/**
 * Item
 * Encapsulates items in the library collection
 */
public class Item {
  private Integer itemID;
  private String title;
  private String author;
  private Borrower borrowedBy;

  /**
   * Constructor for Item with given item ID, title, author
   */
  public Item(Integer iID, String ttl, String aut)
                throws IllegalArgumentException {
   if (!dataValid(iID, ttl, aut)) {
     throw new IllegalArgumentException("Bad arg(s) in Item constructor");
   }
   borrowedBy = null; //initially, not out on loan
   itemID = iID;
   title = ttl;
   author = aut;
  }

  public Integer getItemID() { return itemID; }

  public Borrower getBorrowedBy()          { return borrowedBy; }
  public void     setBorrowedBy(Borrower b) { borrowedBy = b;    }
```

```
  public void     clearBorrowedBy()        { borrowedBy = null; }

  public String getTitle()  { return title;  }
  public String getAuthor() { return author; }

  public String toString() {
    if (borrowedBy != null)
      return String.format("%05d: %s by %s: \non loan to borrower %s",
          itemID, title, author, borrowedBy);
    else
      return String.format("%05d: %s by %s: not on loan", itemID, title, author);
  }

  /**
   * Test validity of Item data:
   *   title, author are strings of length at least 1
   *   ID is not negative.
   */
  public static boolean dataValid(
            int id, String ttl, String aut) {
    if (ttl.length() == 0) return false;
    if (aut.length() == 0) return false;
    if (id < 0) return false;
    return true;
  }
}
```