

Course: Principals of Software Development – ENSF 409

Lab 7

Instructor: M. Moshirpour

Student Name: Mitchell Sawatzky

Date Submitted: March 11, 2016

Exercise A

MusicRecord.java

```
import java.io.Serializable;

/**
 * A simple class that represents a music record
 *
 */
class MusicRecord implements Serializable
{
    private int year_recorded;
    private String songName;
    private String singerName;
    private double purchase_price;
    private static final long serialVersionUID = 1L;

    /**
     * A default constructor that builds a record with blank data
     */
    public MusicRecord() {
        this( 0, "", "", 0.0 );
    }

    /**
     * A constructor that initializes the music records with supplied
     * data.
     */
    public MusicRecord( int year, String song, String singer, double value ) {
        setYear( year );
        setSongName( song );
        setSingerName( singer );
        setPrice( value );
    }

    /**
     * sets the data field year_recorded to supplied argument year
     * data.
     */
    public void setYear( int year ) {
        year_recorded = year;
    }
}
```

```

/**
 * Returns the recording year
 */
    public int getYear() {
        return year_recorded;
    }

/**
 * sets the data field songName to supplied argument song
 */
    public void setSongName( String song ) {
        songName = song;
    }

/**
 * Returns the songName name
 */
    public String getSongName() {
        return songName;
    }

/**
 * sets the data field sinterName to supplied argument singer
 */
    public void setSingerName( String singer ) {
        singerName = singer;
    }

/**
 * Returns the singer's name
 */
    public String getSingerName() {
        return singerName;
    }

/**
 * sets the data field purchase_price to supplied argument price
 */
    public void setPrice( double value ) {

```

```

        purchase_price = value;
    }

    /**
     * Returns the price
     */
    public double getPurchasePrice(){
        return purchase_price;
    }
}

```

ReadRecord.java

```

/**
 * Started by M. Moussavi
 * March 2016
 * Completed by: Mitchell Sawatzky
 */

import java.io.EOFException;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;

public class ReadRecord {

    private ObjectInputStream input;

    /**
     * opens an ObjectInputStream using a FileInputStream
     */

    private void readObjectsFromFile(String name)
    {
        MusicRecord record;

        try
        {
            input = new ObjectInputStream(new FileInputStream( name ) );
        }
        catch ( IOException ioException )
        {
            System.err.println( "Error opening file." );
        }
    }
}

```

```

    }

    /* The following loop is supposed to use readObject method of of
    * ObjectInputStream to read a MusicRecord object from a binary file that
    * contains several records.
    * Loop should terminate when an EOFException is thrown.
    */

    try {
        while (true) {
            record = (MusicRecord)input.readObject();
            System.out.println(record.getYear() + "\t" + record.getSongName() + "\t" +
record.getSingerName() + "\t" + record.getPurchasePrice());
        }
    } catch (EOFException e) {
    } catch (ClassNotFoundException e) {
        System.err.println("Could not find the class associated with this serialization");
        System.err.println(e.getStackTrace());
    } catch (IOException e) {
        System.err.println("Could not read object");
        System.err.println(e);
    }

    try {
        input.close();
    } catch (IOException e) {
        System.err.println("Could not close file");
        System.err.println(e.getStackTrace());
    }
}

public static void main(String [] args)
{
    ReadRecord d = new ReadRecord();
    d.readObjectsFromFile("allsongs.ser");
}
}

```

WriteRecord.java

```
/**
```

```

* Started by M. Moussavi
* March 2016
* Completed by: Mitchell Sawatzky
*/

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.util.NoSuchElementException;
import java.util.Scanner;

public class WriteRecord {

    ObjectOutputStream objectOut = null;
    MusicRecord record = null;
    Scanner stdin = null;
    Scanner textFileIn = null;

    /**
     * Creates an blank MusicRecord object
     */
    public WriteRecord() {
        record = new MusicRecord();
    }

    /**
     * initializes the data fields of a record object
     * @param year - year that song was purchased
     * @param songName - name of the song
     * @param singerName - singer's name
     * @param price - CD price
     */
    public void setRecord(int year, String songName, String singerName,
                           double price) {

        record.setSongName(songName);
        record.setSingerName(singerName);
        record.setYear(year);
        record.setPrice(price);
    }
}

```

```

/**
 * Opens a file input stream, using the data field textFileIn
 * @param textFileName name of text file to open
 */
public void openFileInputStream(String textFileName) {
    try {
        textFileIn = new Scanner(new FileInputStream(textFileName));
    } catch (FileNotFoundException e) {
        System.err.println("Could not open " + textFileName);
        System.err.println(e.getStackTrace());
        System.exit(1);
    }
}

/**
 * Opens an ObjectOutputStream using objectOut data field
 * @param objectFileName name of the object file to be created
 */
public void openObjectOutputStream(String objectFileName) {
    try {
        objectOut = new ObjectOutputStream(new FileOutputStream(objectFileName));
    } catch (IOException e) {
        System.err.println("Could not open " + objectFileName);
        System.err.println(e.getStackTrace());
        System.exit(1);
    }
}

/**
 * Reads records from given text file, fills the blank MusicRecord
 * created by the constructor with the existing data in the text
 * file and serializes each record object into a binary file
 */
public void createObjectFile() {

    while (textFileIn.hasNext()) // loop until end of text file is reached
    {

        int year = Integer.parseInt(textFileIn.nextLine());
        System.out.print(year + " "); // echo data read from text file

        String songName = textFileIn.nextLine();
        System.out.print(songName + " "); // echo data read from text file
    }
}

```

```

        String singerName = textFileIn.nextLine();
        System.out.print(singerName + " "); // echo data read from text file

        double price = Double.parseDouble(textFileIn.nextLine());
        System.out.println(price + " "); // echo data read from text file

        setRecord(year, songName, singerName, price);
        textFileIn.nextLine(); // read the dashed lines and do nothing

        try {
            objectOut.writeObject(record);
            objectOut.reset();
        } catch (IOException e) {
            System.err.println("Could not write object file");
            System.err.println(e.getStackTrace());
        }
    }
    try {
        objectOut.close();
    } catch (IOException e) {
        System.err.println("Error closing file");
    }
}

public static void main(String[] args) throws IOException {

    WriteRecord d = new WriteRecord();

    String textFileName = "someSongs.txt"; // Name of a text file that contains
                                           // song records

    String objectFileName = "mySongs.ser"; // Name of the binary file to
                                           // serialize record objects

    d.openFileInputStream(textFileName); // open the text file to read from

    d.openObjectOutputStream(objectFileName); // open the object file to
                                           // write music records into it

    d.createObjectFile(); // read records from opened text file, and write

```



```
        // them into the object file.  
    }  
}
```

Terminal output

```
Mitchell@ttys000 15:54 {0} [exa]$ java ReadRecord  
1999    I have a dream  ABBA      23.5  
2000    S.O.S.    ABBA      14.5  
2013    WaterlooABBA    30.0  
2012    Dream OnAerosmith      12.0  
2008    Grace like rain Todd Agnew      6.7  
1999    Old soulThea Gilmor      4.5  
1988    Honey Bunny      Adam Gilmor      6.7  
2000    Poison Arrow      ABC      7.95  
1988    How Long?      Ace      12.45  
1991    Straight From the Heart Bryan Adams      9.9  
1999    Rolling in the deep      Adele      11.98  
1999    I see stars      Cindy Alxander      23.5  
2000    I am Eighteen      Alice Cooper      14.5  
2013    WaterlooABBA    30.0  
2012    Smile      Lily Allen      12.0  
2008    Tear in your hand      Tori Amos      6.7  
1999    Crash about to happen      Brett Anderson      4.5  
1988    O Superman      Laurie Anderson      6.7  
2000    Do-Re-MiJulie Andrews      7.95
```

Exercise B

BinSearchTree.java

```
import java.io.*;  
import javax.swing.JOptionPane;  
/**  
 * The following class called BinSearchTree creates an empty  
 * tree, and allows the user of the class to insert new nodes into the tree.  
 */  
  
class BinSearchTree {  
  
    Node root;  
  
    /* the following constructor creates an empty tree. */  

```

```

public BinSearchTree() {
    root = null;
}

/**
 * the following method, inserts a new node that contains several data
 * fields, the student's id, faculty, major, and year into the list.
 * @param id - student's id number
 * @param faculty - faculty code
 * @param major - student's major
 * @param year - student's year of study
 */
public void insert(String id, String faculty, String major, String year) {

    Node node = new Node(id, faculty, major, year);

    if(root == null)
        root = node;
    else{
        Node cur;
        // find the location to insert a new node.
        cur = search(root, node);
        // the following if...else block attaches the new node to the left or right wing.

        if(cur.data.id.length() > node.data.id.length())
            cur.left = node;
        else if (cur.data.id.length() < node.data.id.length())
            cur.right = node;
        else if (cur.data.id.compareTo(node.data.id) < 0)
            cur.right = node;
        else if (cur.data.id.compareTo(node.data.id) > 0)
            cur.left = node;
        // if data already exists, send an error message to the user.
        else {
            JOptionPane.showMessageDialog(null, "\nCannot insert: data already " +
                " exists: \n" + id, "
Warning", JOptionPane.PLAIN_MESSAGE);
            node = null;
        }
    }
}

```

```

/**
 * Returns true if tree is empty
 */

public boolean empty() {
    return (root == null);
}

/**
 * removes all nodes
 */

public void destroy() {
//    splice();
    root = null;
}

public void splice() {
    if(!empty()){

        if (root.left!=null){
            root = root.left;
            splice();
        }
        root.left = null;
        if (root.right!=null){
            root = root.right;
            splice();
        }
        root.right = null;
    }
}

/**
 * the following method finds and returns a reference to a node with the
 * target id or returns null, if fails to find a node containing target id.
 * @param start - starting point of the tree (root node)
 * @param target_id - id number that is searching for
 * @return - node that matches its id with the target id. Otherwise returns null.
 */
public Node find( Node start, String target_id) {

```

```

        if(start == null) return null;
        if(start.data.id.equals(target_id)) return start;
        if(target_id.compareTo(start.data.id)>0)
            return find(start.right, target_id);
        else if(target_id.compareTo(start.data.id)<0)
            return find(start.left, target_id);
        return null;
    }

    /**
     * the following function returns a reference to the node that the new node
     * must be attached to. Or returns null if such a node does not exist.
     * @param cur - current node that the new node should be attached to
     * @param node - the new node
     * @return - node that matches its id with the target id. Otherwise returns null.
     */
    private Node search(Node cur, Node node) {
        if((cur.left ==null && cur.right ==null)||
            ((node.data.id.length()< cur.data.id.length()) && (cur.left==null))||
            ((node.data.id.length()> cur.data.id.length()) && (cur.right==null)))
            return cur;
        if((node.data.id.length() < cur.data.id.length()) && (cur.left !=null))
            cur =search(cur.left, node);
        else if((node.data.id.length()>cur.data.id.length()) && (cur.right != null))
            cur =search(cur.right, node);
        else if((node.data.id.compareTo( cur.data.id)>0) && cur.right!=null)
            cur=search(cur.right, node);
        else if((node.data.id.compareTo( cur.data.id)<0) && cur.left!=null)
            cur=search(cur.left, node);
        return cur;
    }

    /**
     * function tha recursively display the data.
     * @param cur - current node
     * @param out - the output stream to print the node information
     * @throws IOException
     */
    public void print_tree(Node cur, PrintWriter out) throws IOException {
        if (cur.left!=null)
            print_tree(cur.left,out);

        String s = cur.data.id+"          "+cur.data.faculty+"          "+

```

```

        cur.data.major+ "          "+cur.data.year+"\n";

        out.println(s);
        System.out.println(s);
        if (cur.right!=null)
            print_tree(cur.right,out);
    }
}

```

Data.java

```

/**
 * Represents data within a node
 */
public class Data {

    /**
     * id is the student's id
     * faculty is the student's faculty
     */
    String id,faculty, major, year;

    public Data( String i, String f, String m, String y)
    {
        id = i;
        faculty = f;
        major = m;
        year = y;
    }

    public String toString()
    {
        return ("id : " + id + " faculty: " + faculty + " major: " + major +
                " year: " + year);
    }
}

```

GUI.java

```

import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.Scanner;

```

```

import java.io.IOException;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.io.FileOutputStream;
import java.io.FileInputStream;

/**
 * Container class for the GUI
 * @author Mitchell Sawatzky
 * @version 1.0
 * @since Mar 2016
 */
public class GUI {
    /**
     * True if the a BST has been generated from a file
     */
    static boolean generated = false;

    /**
     * The path to the file the BST was generated from
     */
    static String path;

    /**
     * Program entry point
     * @param argv - cli arguments
     */
    public static void main (String[] argv) {
        // Declare all top-level frame elements
        JFrame f = new JFrame("Exercise B");
        JPanel p = new JPanel();
        JPanel buttons = new JPanel();
        JButton insert = new JButton("Insert");
        JButton query = new JButton("Query");
        JButton generate = new JButton("Generate New Tree");
        JTextArea t = new JTextArea();
        JScrollPane wrapper;

        // Declare BST
        BinSearchTree bst = new BinSearchTree();

        // register listener for the insert button
        insert.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

```

```

        if (generated) {
            // declare input fields
            JTextField id = new JTextField();
            JTextField faculty = new JTextField();
            JTextField major = new JTextField();
            JTextField year = new JTextField();
            Object[] inp = {
                "ID:", id,
                "Faculty:", faculty,
                "Major:", major,
                "Year:", year
            };

            // ask for inputs
            int o = JOptionPane.showConfirmDialog(null, inp, "Add Record",
JOptionPane.OK_CANCEL_OPTION);

            // check if the user hit ok or cancel
            if (o == JOptionPane.OK_OPTION) {
                // make sure ID is valid
                try {
                    int t = Integer.parseInt(id.getText());
                } catch (NumberFormatException err) {
                    JOptionPane.showMessageDialog(null, "Field 'ID' must be a number", "Cannot
insert into tree", JOptionPane.PLAIN_MESSAGE);
                    return;
                }
                // make sure YEAR is valid
                try {
                    int t = Integer.parseInt(year.getText());
                } catch (NumberFormatException err) {
                    JOptionPane.showMessageDialog(null, "Field 'Year' must be a number", "Cannot
insert into tree", JOptionPane.PLAIN_MESSAGE);
                    return;
                }
                // write changes to the file
                try {
                    PrintWriter writer = new PrintWriter(new FileOutputStream(path, true));
                    writer.append("      " + id.getText() + "      " + faculty.getText() + "      " +
major.getText() + "      " + year.getText() + "\n");
                    writer.close();
                } catch (IOException err) {
                    JOptionPane.showMessageDialog(null, "Could not save new record to file",
"Internal Error", JOptionPane.PLAIN_MESSAGE);

```

```

        return;
    }
    // update BST
    bst.insert(id.getText(), faculty.getText(), major.getText(), year.getText());
    //update the text area
    try {
        updateText(bst, t);
    } catch (IOException err) {
        JOptionPane.showMessageDialog(null, "Could not turn BST into text", "Internal
Error", JOptionPane.PLAIN_MESSAGE);
        System.err.println(e);
        return;
    }
}
} else {
    // the user has not generated a tree yet
    JOptionPane.showMessageDialog(null, "Please generate a tree first", "Cannot Insert
Into Tree", JOptionPane.PLAIN_MESSAGE);
}
}
});

// register listener for the query button
query.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (generated) {
            // search for the id
            Node res = bst.find(bst.root, JOptionPane.showInputDialog(null, "Student ID:", "Input
Student ID", JOptionPane.PLAIN_MESSAGE));
            // display results
            if (res != null) {
                JOptionPane.showMessageDialog(null, res.toString(), "Query Result",
JOptionPane.PLAIN_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Could not find a student with that ID",
"Query Result", JOptionPane.PLAIN_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(null, "Please generate a tree first", "Cannot Query
Tree", JOptionPane.PLAIN_MESSAGE);
        }
    }
});

```



```

// register listener for for the generate button
generate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // ask for path
        path = JOptionPane.showInputDialog(null, "File Path", "Input File Name",
JOptionPane.PLAIN_MESSAGE);
        // declare line counter
        int line = 1;
        try {
            // open file and iterate over each line
            Scanner in = new Scanner(new FileInputStream(path));
            while (in.hasNextLine()) {
                String record = in.nextLine();
                // trim the line of whitespace and split it every time there is one or more spaces
                String[] rec = record.trim().split(" ");
                // check that there are a correct number of fields in the line
                if (rec.length != 4) {
                    JOptionPane.showMessageDialog(null, "Incomplete record at " + path + " (line "
+ line + ")", "Cannot Generate Tree", JOptionPane.PLAIN_MESSAGE);
                    return;
                }
                // update the tree
                bst.insert(rec[0], rec[1], rec[2], rec[3]);
                line++;
            }
            // close the file
            in.close();
        } catch (IOException err) {
            JOptionPane.showMessageDialog(null, "Could not file file: " + path, "Cannot Generate
Tree", JOptionPane.PLAIN_MESSAGE);
            return;
        }

        // write bst data to the textarea
        try {
            updateText(bst, t);
        } catch (IOException err) {
            JOptionPane.showMessageDialog(null, "Could not turn BST into text", "Internal Error",
JOptionPane.PLAIN_MESSAGE);
            System.err.println(e);
            return;
        }
        // enable use of other buttons

```

```

        generated = true;
    }
});

// set the display area as non-editable
t.setEditable(false);
// allow the display area to scroll if there is overflow
wrapper = new JScrollPane(t);

// add buttons to the button panel
buttons.add(insert);
buttons.add(query);
buttons.add(generate);

// add content into the main panel
p.setLayout(new BorderLayout());
p.add("North", buttons);
p.add("Center", wrapper);

// add the main panel to the main frame
f.add(p);
// show the main frame
f.setVisible(true);
// set the size of the main frame
f.setSize(500,500);
// configure the behaviour of the exit button
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

/**
 * Update the TextArea with the content of the bst
 * @throws IOException
 * @param bst - the BinSearchTree to grab data from
 * @param t - the JTextArea to write the data to
 */
public static void updateText (BinSearchTree bst, JTextArea t) throws IOException {
    // a buffer for the printwriter to write to
    StringWriter buf = new StringWriter();
    // the printwriter required by BinSearchTree.print_tree
    PrintWriter writer = new PrintWriter(buf);
    bst.print_tree(bst.root, writer);
    // write the contents of the buffer to the TextArea

```

```
        t.setText(buf.toString());
    }
}
```

Node2.java

```
/**
 * A node with data and two references to the next-left and next-right node
 */
class Node {

    Data data;
    Node left, right;

    /**
     *
     * @param id student id
     * @param faculty faculty code
     * @param major student's major
     * @param year student's year of study
     */
    public Node(String id, String faculty, String major, String year) {
        // creating a data item
        data = new Data(id, faculty, major, year);
        left = null;
        right = null;
    }

    public String toString() {
        return data.toString();
    }
}
```

Exercise B

Insert

Query

Generate New Tree

Input File Name

File Path

Cancel

OK

exB

Mitchell@ttys000 15:47 {0} [exb]\$ java gui
Error: Could not find or load main class gui
Mitchell@ttys000 15:47 {1} [exb]\$ java GUI

Exercise B


Insert

Query

Generate New Tree

| | | | |
|-------|----|------|---|
| 64568 | EN | ENEL | 1 |
| 64939 | EN | ENCH | 2 |
| 64971 | EN | ENOG | 2 |
| 65136 | EN | ENGO | 2 |
| 65387 | EN | ENGO | 2 |
| 66002 | EN | ENMF | 2 |
| 66268 | EN | ENCI | 2 |
| 66408 | EN | ENEL | 1 |
| 66472 | EN | ENGO | 3 |
| 66600 | EN | ENCI | 2 |
| 69887 | EN | ENCH | 2 |
| 70080 | EN | ENME | 2 |
| 70082 | EN | ENSF | 4 |

Add Record



ID:
70084

Faculty:
EN

Major:
ENSF

Year:
2

Cancel

OK

exB

| | | | |
|-------|----|------|---|
| 64971 | EN | ENOG | 2 |
| 65136 | EN | ENGO | 2 |
| 65387 | EN | ENGO | 2 |
| 66002 | EN | ENMF | 2 |
| 66268 | EN | ENCI | 2 |
| 66408 | EN | ENEL | 1 |
| 66472 | EN | ENGO | 3 |
| 66600 | EN | ENCI | 2 |
| 69887 | EN | ENCH | 2 |
| 70080 | EN | ENME | 2 |
| 70082 | EN | ENSF | 4 |

Exercise B

Insert

Query

Generate New Tree

| | | | |
|-------|----|------|---|
| 64568 | EN | ENEL | 1 |
| 64939 | EN | ENCH | 2 |
| 64971 | EN | ENOG | 2 |
| 65136 | EN | ENGO | 2 |
| 65387 | EN | ENGO | 2 |
| 66002 | EN | ENMF | 2 |
| 66268 | EN | ENCI | 2 |
| 66408 | EN | ENEL | 1 |
| 66472 | EN | ENGO | 3 |
| 66600 | EN | ENCI | 2 |
| 69887 | EN | ENCH | 2 |
| 70080 | EN | ENME | 2 |
| 70082 | EN | ENSF | 4 |

exB

| | | | |
|-------|----|------|---|
| 64971 | EN | ENOG | 2 |
| 65136 | EN | ENGO | 2 |
| 65387 | EN | ENGO | 2 |
| 66002 | EN | ENMF | 2 |
| 66268 | EN | ENCI | 2 |
| 66408 | EN | ENEL | 1 |
| 66472 | EN | ENGO | 3 |
| 66600 | EN | ENCI | 2 |
| 69887 | EN | ENCH | 2 |
| 70080 | EN | ENME | 2 |
| 70082 | EN | ENSF | 4 |

Exercise B

Insert

Query

Generate New Tree

| | | | |
|-------|----|------|---|
| 64939 | EN | ENCH | 2 |
| 64971 | EN | ENOG | 2 |
| 65136 | EN | ENGO | 2 |
| 65387 | EN | ENGO | 2 |
| 66002 | EN | ENMF | 2 |
| 66268 | EN | ENCI | 2 |
| 66408 | EN | ENEL | 1 |
| 66472 | EN | ENGO | 3 |
| 66600 | EN | ENCI | 2 |
| 69887 | EN | ENCH | 2 |
| 70080 | EN | ENME | 2 |
| 70082 | EN | ENSF | 4 |
| 70084 | EN | ENSF | 2 |

Input Student ID

Student ID:

64971

Cancel

OK

exB

| | | | |
|-------|----|------|---|
| 65136 | EN | ENGO | 2 |
| 65387 | EN | ENGO | 2 |
| 66002 | EN | ENMF | 2 |
| 66268 | EN | ENCI | 2 |
| 66408 | EN | ENEL | 1 |
| 66472 | EN | ENGO | 3 |
| 66600 | EN | ENCI | 2 |
| 69887 | EN | ENCH | 2 |
| 70080 | EN | ENME | 2 |
| 70082 | EN | ENSF | 4 |
| 70084 | EN | ENSF | 2 |

Exercise B

Insert

Query

Generate New Tree

| | | | |
|-------|----|------|---|
| 64939 | EN | ENCH | 2 |
| 64971 | EN | ENOG | 2 |
| 65136 | EN | ENGO | 2 |
| 65387 | EN | ENGO | 2 |
| 66002 | EN | ENMF | 2 |
| 66268 | EN | ENCI | 2 |
| 66408 | EN | ENEL | 1 |
| 66472 | EN | ENGO | 3 |
| 66600 | EN | ENCI | 2 |
| 69887 | EN | ENCH | 2 |
| 70080 | EN | ENME | 2 |
| 70082 | EN | ENSF | 4 |
| 70084 | EN | ENSF | 2 |

Query Result

id : 64971 faculty: EN major: ENOG year: 2

OK

| | | |
|-------|----|----|
| 65136 | EN | EN |
| 65387 | EN | EN |
| 66002 | EN | EN |
| 66268 | EN | EN |
| 66408 | EN | EN |
| 66472 | EN | EN |
| 66600 | EN | EN |
| 69887 | EN | EN |
| 70080 | EN | EN |
| 70082 | EN | EN |
| 70084 | EN | EN |