รายการอ้างอิง

[1] M. Fowler. Refactoring: Improving the Design of Existing Code. United States: Addison-Wesley, 1999.

[2] Mika Manty la, Jari Vanhanen and Casper Lassenius. A Taxonomy and an Initial Empirical Study of Bad Smell in Code. Proceeding of the International Conference on Software Maintenance (ICSM'03), IEEE Computer Society, 2003.

[3] T. Pienlert and P. Muenchaisri. Bad-Smell Detection using Object-Oriented Software Metrics. International Conference Computer Science, Software Engineering, Information Technology, e-Business and Application (CSITeA '04), December 27-29, 2004.

[4] Radu Marinescu. Detecting Design Flaws via Metric in Object-Oriented System. Proceeding of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and System (TOOLS39), 2003.

[5] Sang-Uk Jeon, Joon-Sang Lee and Doo-Hwan Bae. An Automated Refactoring approach to design pattern-based program transformation in Java programs, Proceedings of the Ninth Asia-Pacific Software Engineering Conference (APSEC'02), IEEE Computer Society, 2002.

[6] M.O Cinneide and P. Nixon. A methodology for the Automated Introduction of Design Patterns, In Proceeding of the IEEE International Conference on Software Maintenance, pp. 463-472, IEEE Press, 1999.

[7] Arthur J. Riel. Object-Oriented Design Heuristics. Addison-Wesley, 1996.

[8] Robert C. Martin, Clean Code. Pearson Education, 2009.

[9] Steven John Metsker, Building Parsers with Java™. Addison-Wesley, 2001.

[10] William C. Wake, Refactoring Workbook. Addison-Wesley, 2003.

[11] บุญเสริม กิจสิริกุล, เอกสารคำสอนวิชา 2110654 (ARTIFICIAL INTELLIGENCE). ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย,15 มี.ค. 2548.

ภาคผนวก

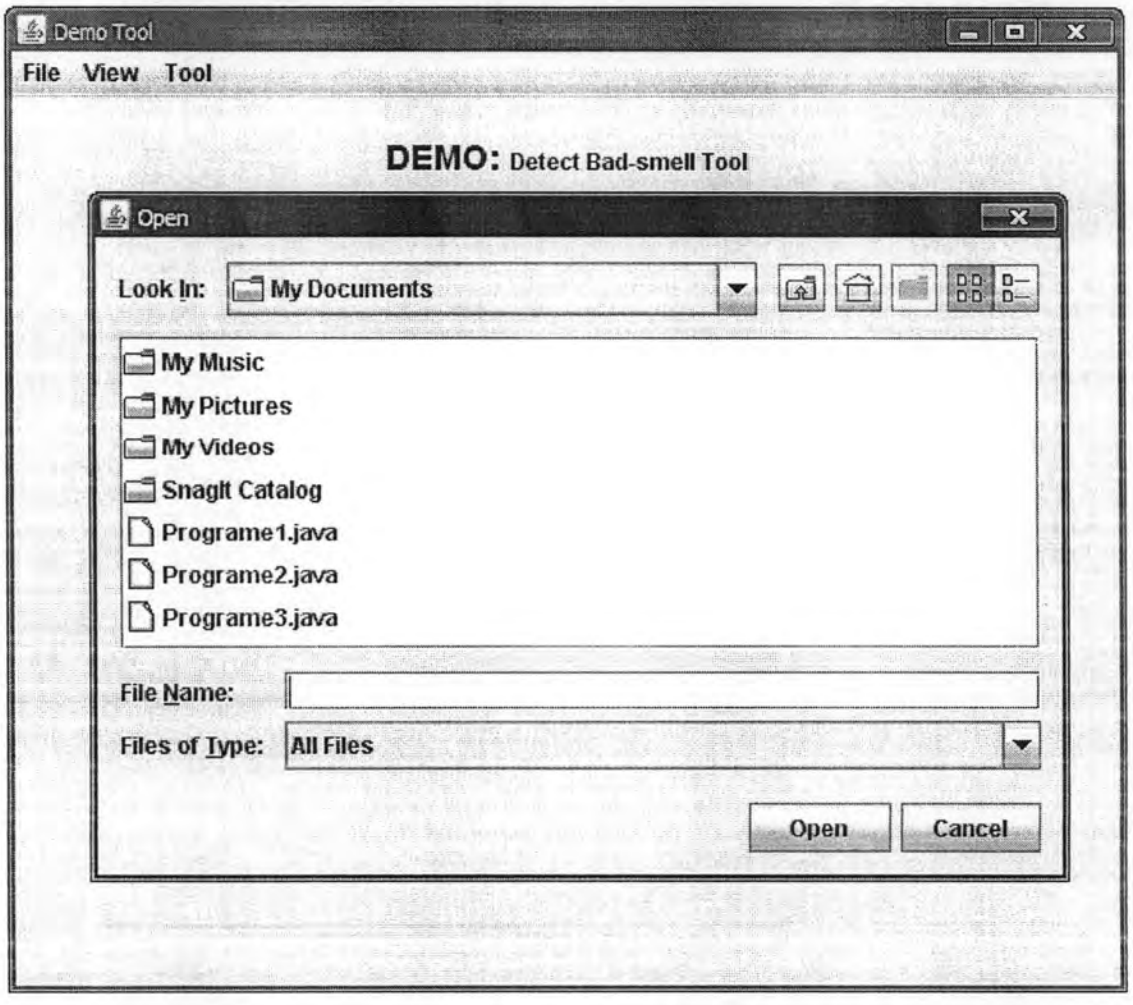
ภาคผนวก ก

การใช้งานเครื่องมือช่วยค้นหาร่องรอยที่ไม่ดี

ในภาคผนวกนี้ จะอธิบายถึงการใช้งานเครื่องมือสำหรับค้นหารองรอยที่ไม่ดี ซึ่งแบ่งออกเป็น 2 ส่วน คือ การอ่านโปรแกรมต้นฉบับ และ การดูผลการค้นหาร่องรอยที่ไม่ดี ดังแสดงรายละเอียดต่อไปนี้

ก.1 การอ่านโปรแกรมต้นฉบับ

เมื่อผู้ใช้เข้าสู่โปรแกรม ผู้ใช้สามารถเลือกโปรแกรมต้นฉบับได้จากเมนู File > Import File หรือ Shift+I เพื่อเลือกโปรแกรมต้นฉบับเข้าสู่โปรเจค ผู้ใช้สามารถเลือกได้ที่ละไฟล์ ดังรูป ก.1



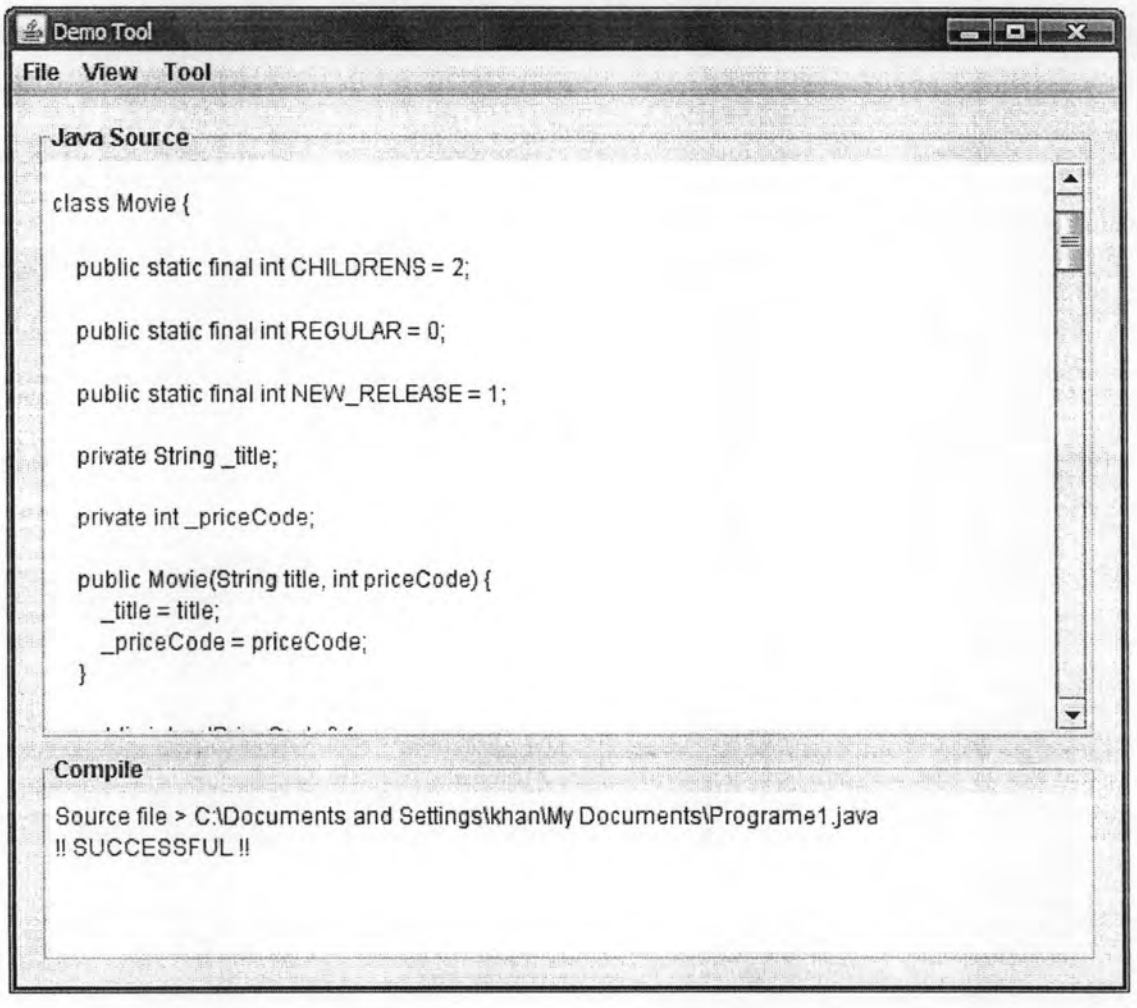
รูปที่ ก.1 แสดงเฟรมที่เลือกโปรแกรมต้นฉบับภาษาจาวา

ก.2 การดูผลการค้นหาร่องรอยที่ไม่ดี

เมื่อผู้ใช้ต้องการดูค่าในการค้นหา ผู้ใช้สามารถเลือกฟังก์ชันดูร่องรอยที่ไม่ดีแบ่งตามประเภทของร่องรอยที่ไม่ดี 4 ประเภท คือ Feature Envy (Attribute) Feature Envy (Method) Message Chains Middle Man และ Inappropriate Intimacy และเลือกดูผลทุกประเภทพร้อมกันได้ มีรายละเอียดดังนี้

ก.2.1 การดูซอร์สโค้ดของโปรแกรมต้นฉบับ

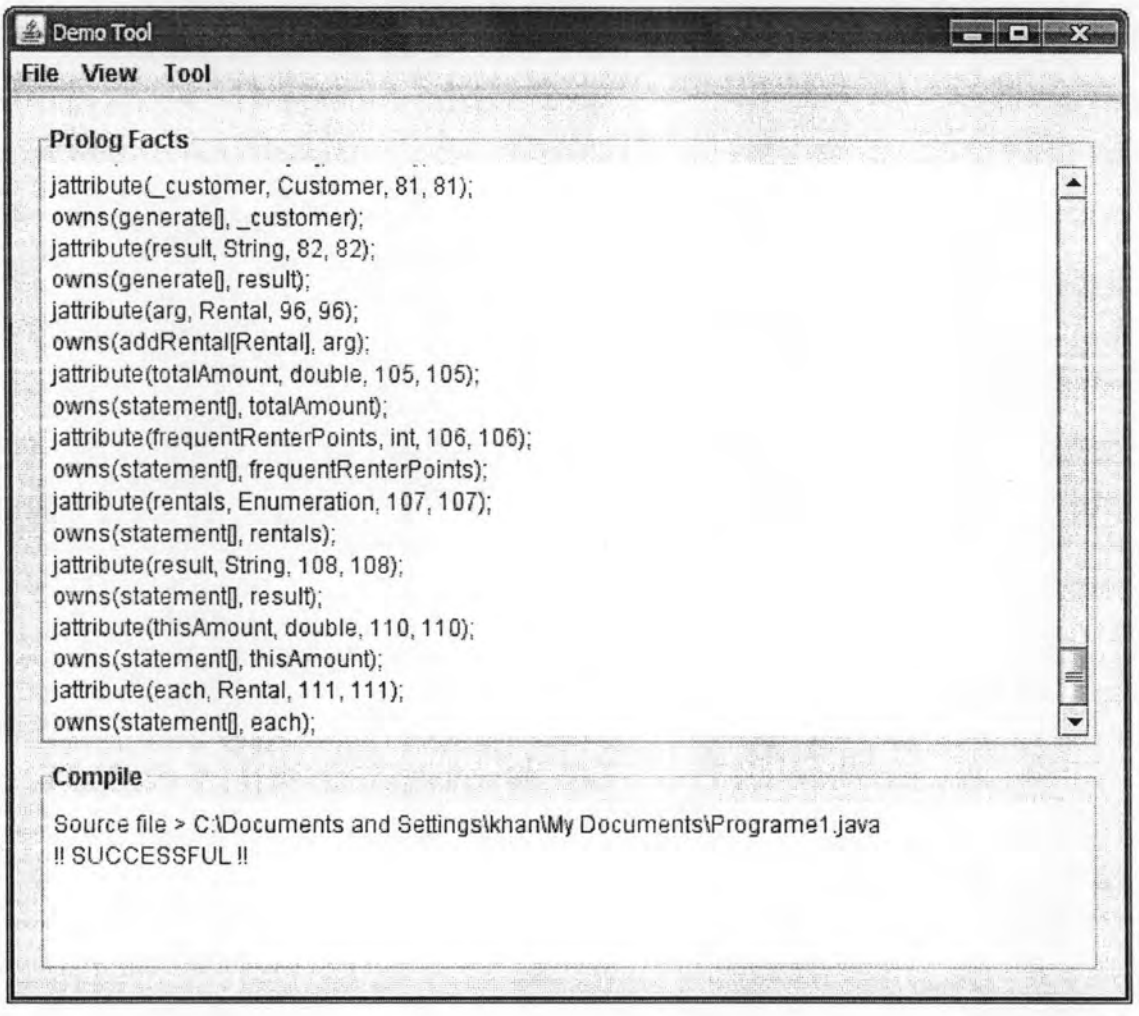
ผู้ใช้สามารถเลือกดูโปรแกรมต้นฉบับที่ใช้ในการค้นหาร่องรอยที่ไม่ดีได้ จาก View > Java หรือ Ctrl+J ระบบก็จะทำการแสดงแฟรมที่มีรายละเอียดของโปรแกรมต้นฉบับ แสดงได้ดังรูปที่ ก.2



รูปที่ ก.2 แสดงเฟรมการดูซอร์สโค้ดของโปรแกรมต้นฉบับ

ก.2.2 การดูโปรล็อก Fact

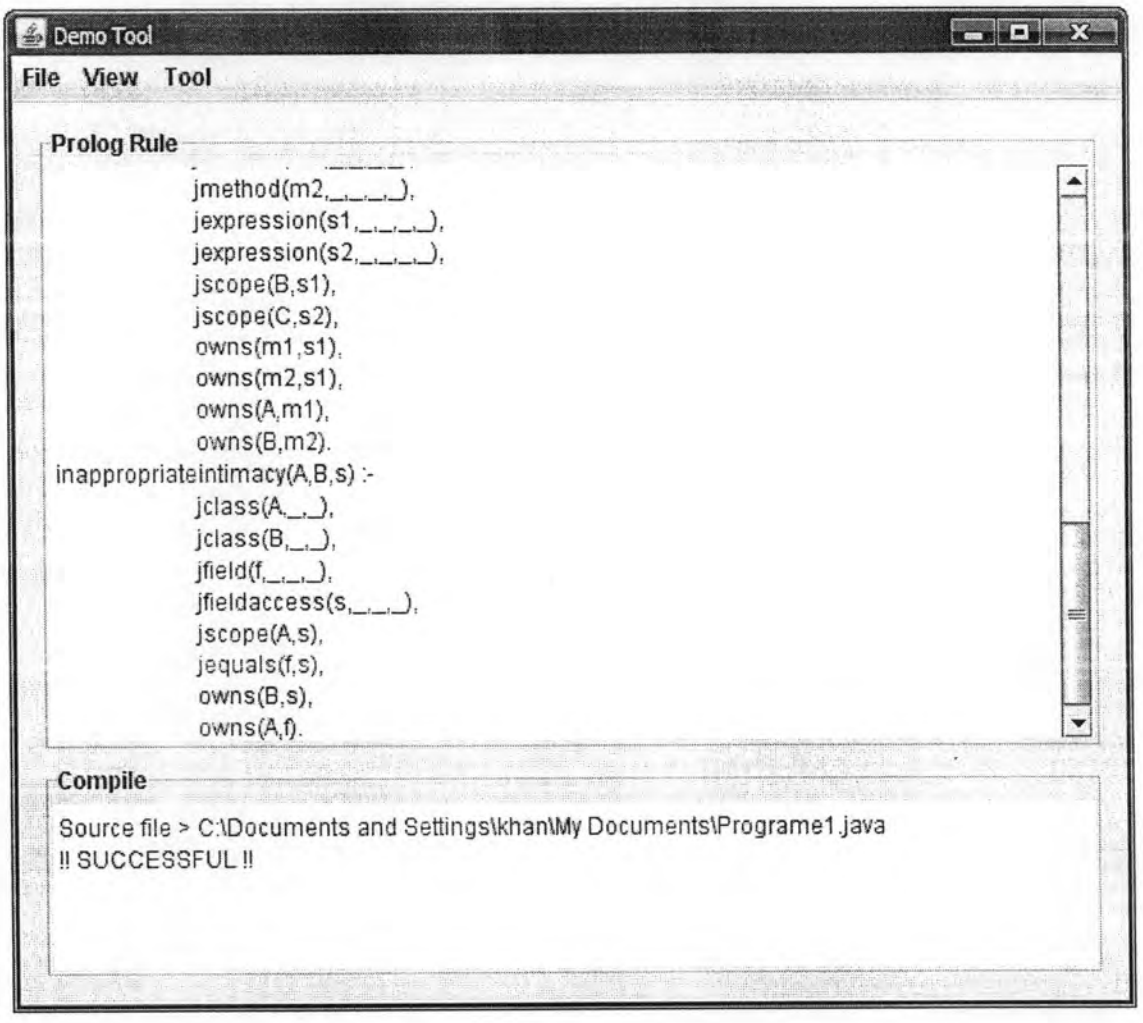
ผู้ใช้สามารถเลือกดูข้อเท็จจริงที่ได้จากการแปลงของโปรแกรมต้นฉบับ ได้จากการเลือกเมนู View > Prolog Facts หรือ Ctrl+F ระบบก็จะทำการแสดงแฟรมที่มีรายละเอียดของข้อเท็จจริง แสดงได้ดังรูปที่ ก.3



รูปที่ ก.3 แสดงเฟรมการดูโปรล็อก Fact

ก.2.3 การดูโปรล็อก Rule

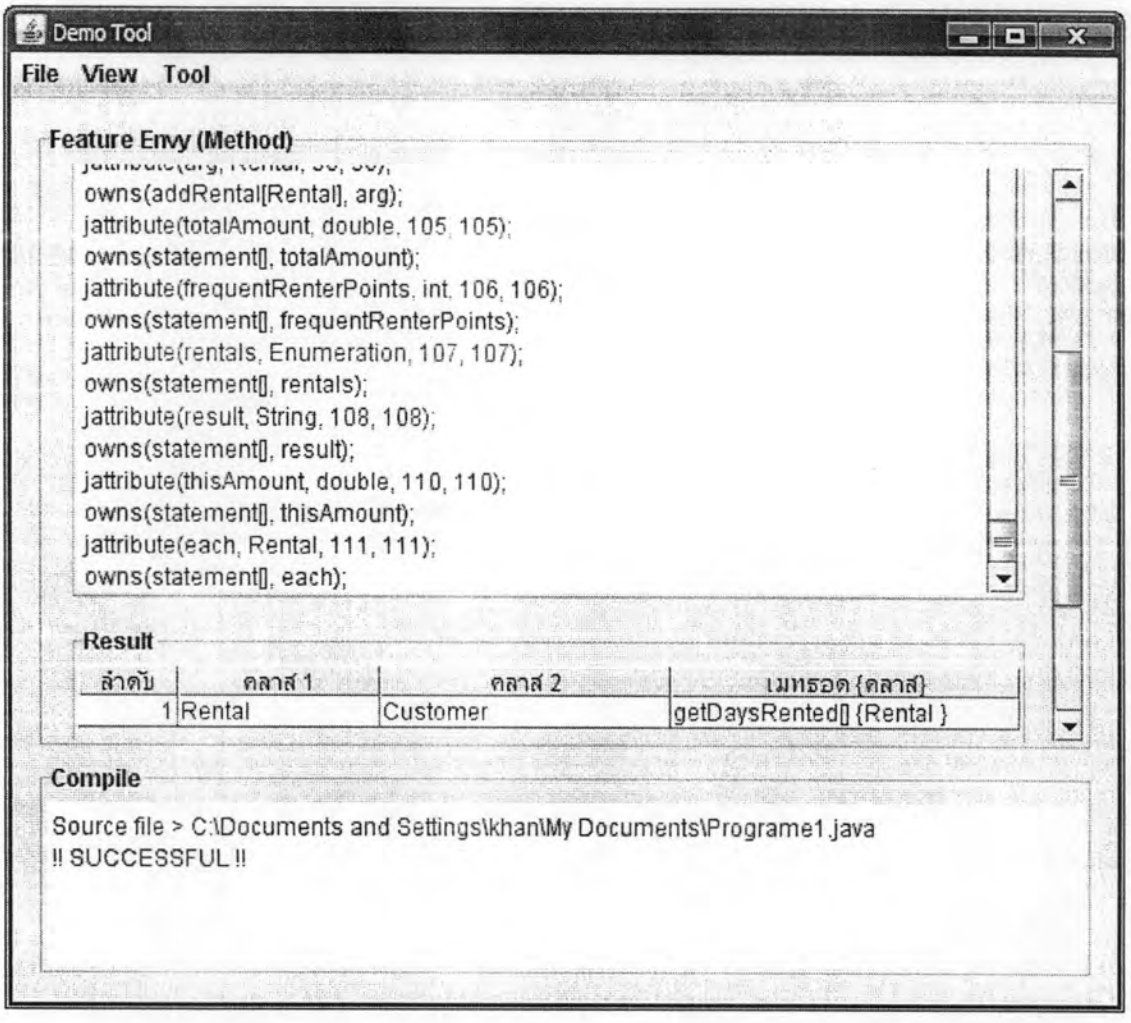
ผู้ใช้สามารถเลือกดูกฎโปรล็อกที่ใช้ในการค้นหา ได้จากการเลือกเมนู View > Prolog Rule หรือ Ctrl+R ระบบก็จะทำการแสดงแฟรมที่มีรายละเอียดของกฎโปรล็อกทั้ง 4 ประเภท แสดงได้ดังรูปที่ ก.4



รูปที่ ก.4 แสดงเฟรมการดูโปรล็อก Rule

ก.2.4 การดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Feature Envy (Method)

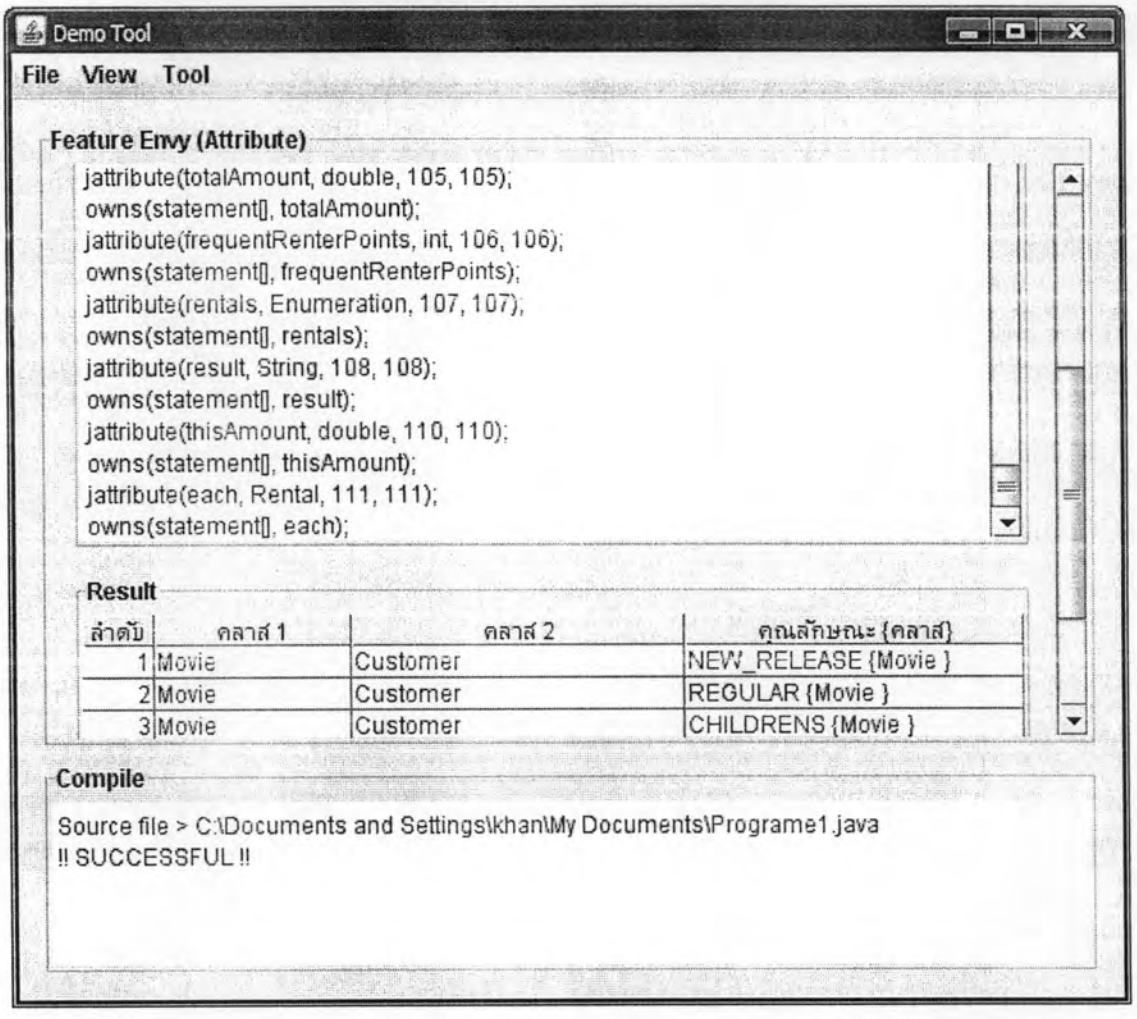
ผู้ใช้สามารถเลือกดูผลการค้นหาร่องรอยที่ไม่ดีประเภท Feature Envy (Method) ได้จากการเลือกเมนู View > MFeature Envy หรือ Shift+M ระบบก็จะทำการแสดงแฟรมที่มีรายละเอียดค่าที่ได้จากการค้นหา แสดงได้ดังรูปที่ ก.5



รูปที่ ก.5 แสดงเฟรมการดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Feature Envy (Method)

ก.2.5 การดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Feature Envy (Attribute)

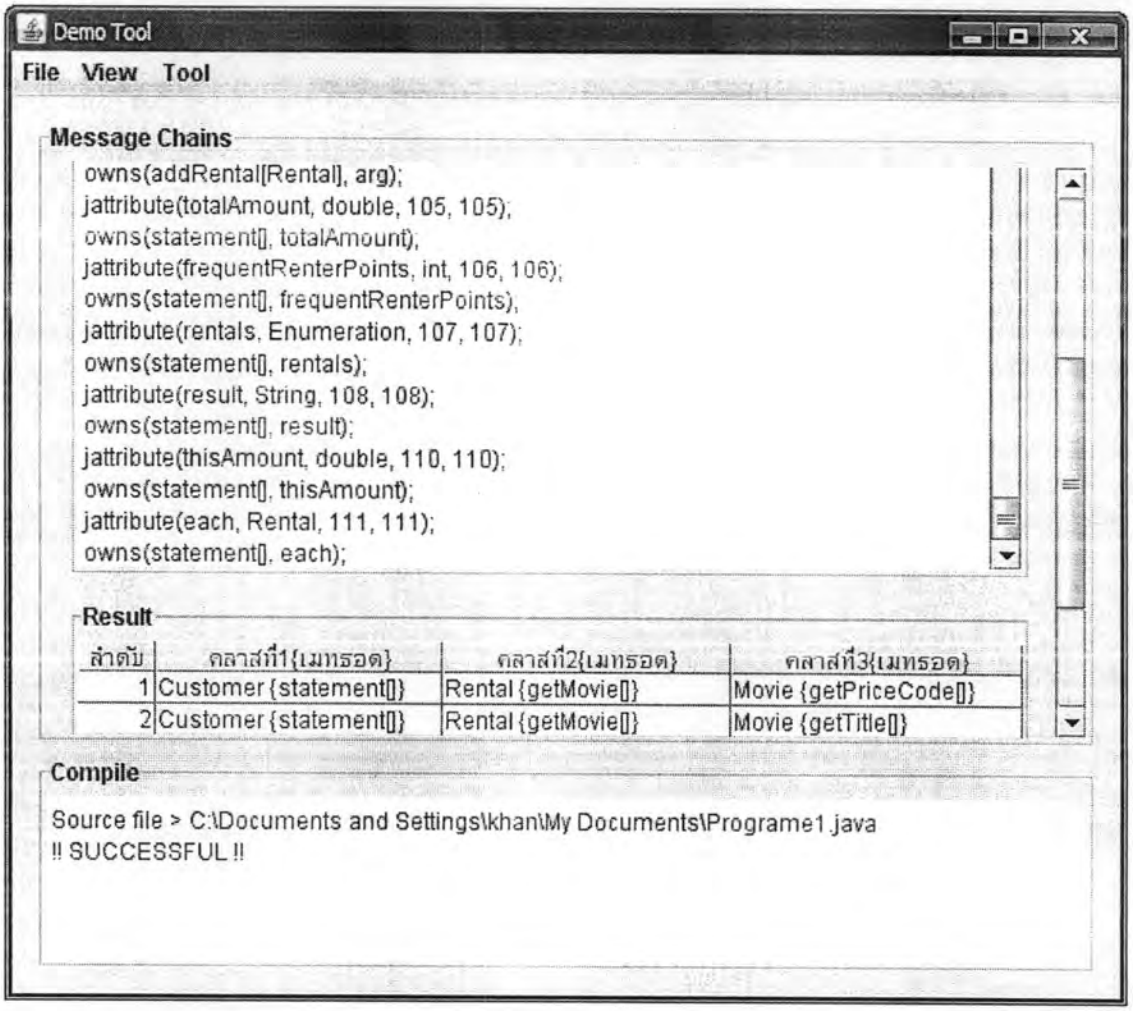
ผู้ใช้สามารถเลือกดูผลการค้นหาร่องรอยที่ไม่ดีประเภท Feature Envy (Attribute) ได้จากการเลือกเมนู View > AFeature Envy หรือ Shift+A ระบบก็จะทำการแสดงแฟรมที่มีรายละเอียดค่าที่ได้จากการค้นหา แสดงได้ดังรูปที่ ก.6



รูปที่ ก.6 แสดงเฟรมการดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Feature Envy (Attribute)

ก.2.6 การดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Message Chains

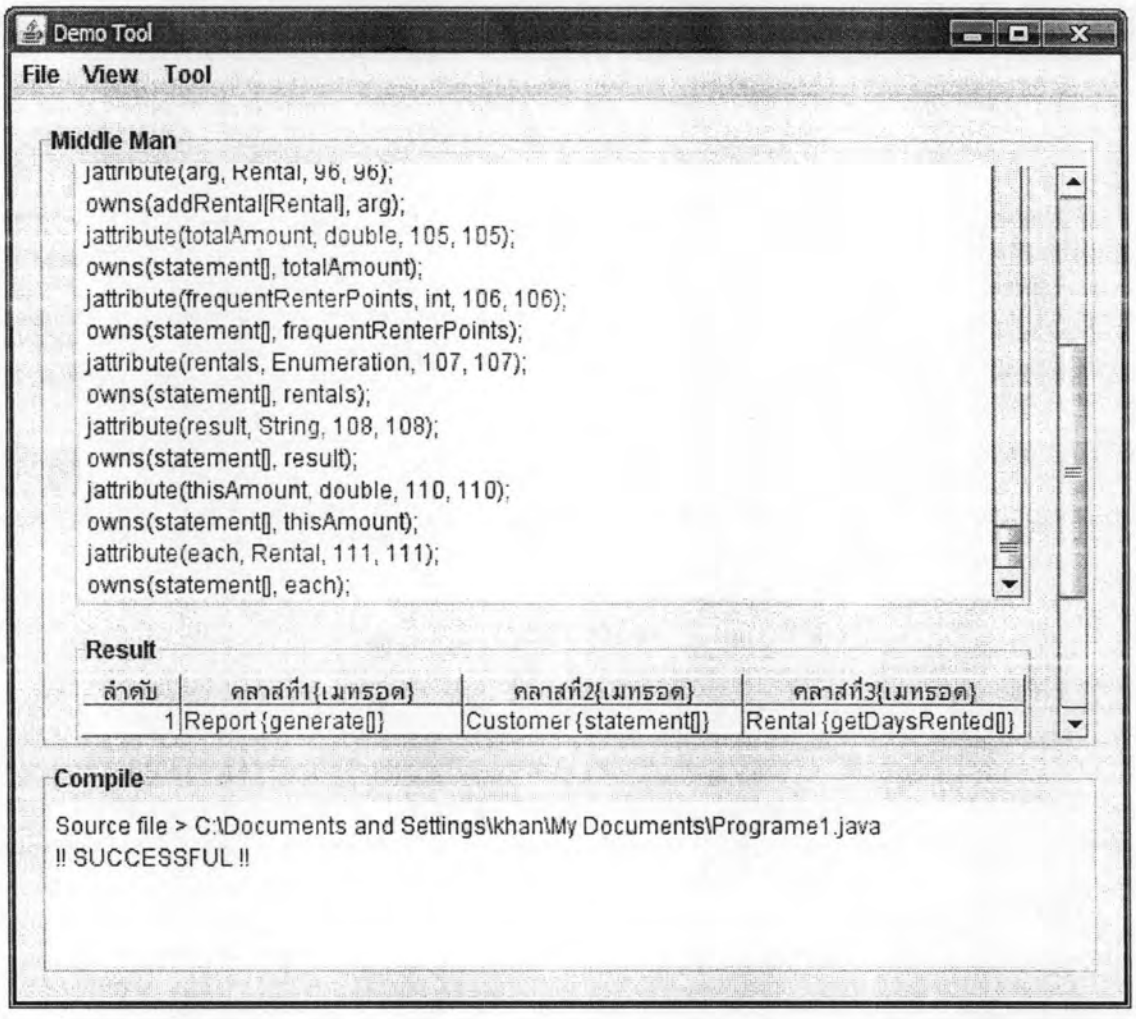
ผู้ใช้สามารถเลือกดูผลการค้นหาร่องรอยที่ไม่ดีประเภท Feature Envy (Method) ได้จากการเลือกเมนู View > Message Chains หรือ Ctrl+C ระบบก็จะทำการแสดงแฟรมที่มีรายละเอียดค่าที่ได้จากการค้นหา แสดงได้ดังรูปที่ ก.7



รูปที่ ก.7 แสดงเฟรมการดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Message Chains

ก.2.7 การดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Middle Man

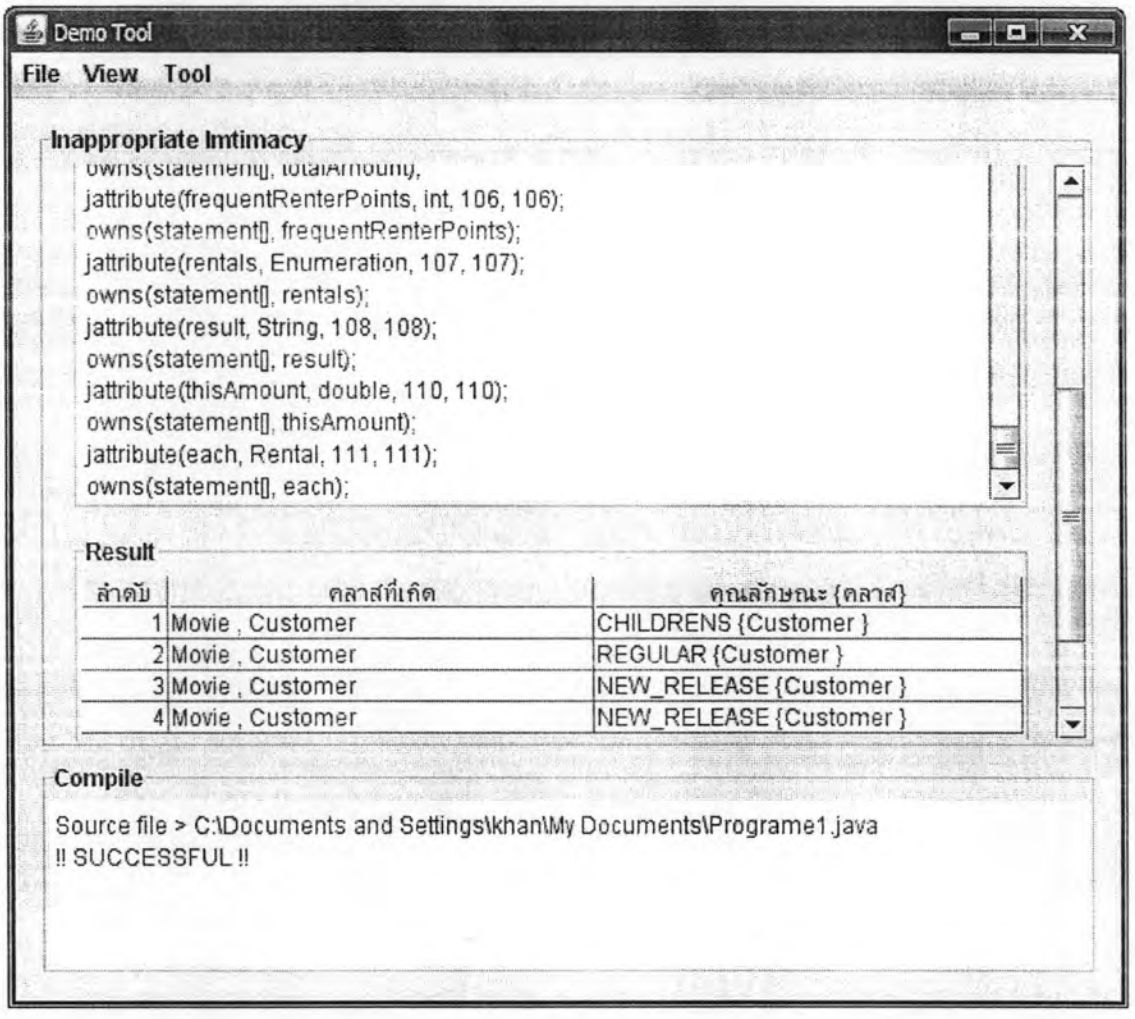
ผู้ใช้สามารถเลือกดูผลการค้นหาร่องรอยที่ไม่ดีประเภท Middle Man ได้จากการเลือกเมนู View > Middle Man หรือ Ctrl+M ระบบก็จะทำการแสดงแฟรมที่มีรายละเอียดค่าที่ได้จากการค้นหา แสดงได้ดังรูปที่ ก.8



รูปที่ ก.8 แสดงเฟรมการดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Middle Man

ก.2.8 การดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Inappropriate Intimacy (General Form)

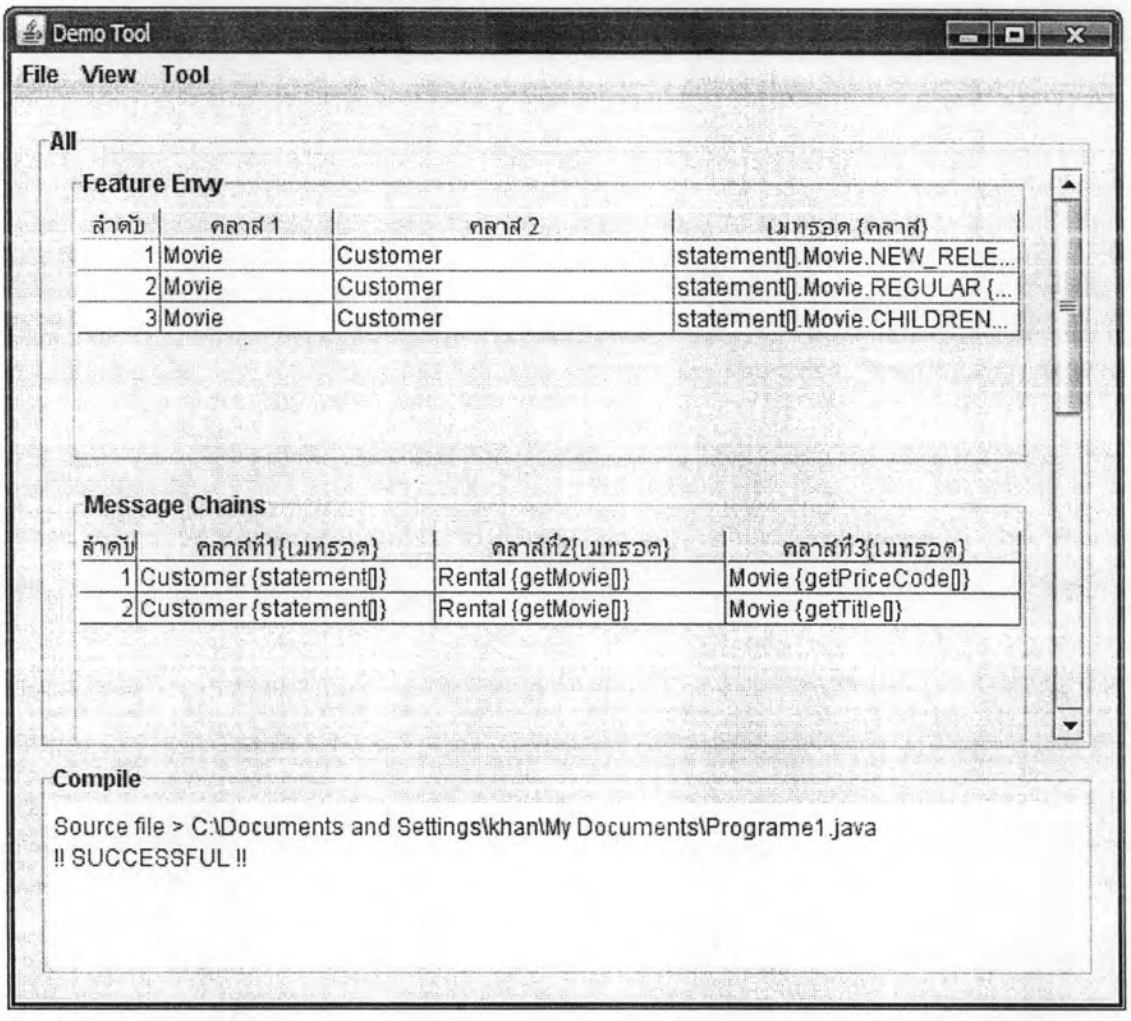
ผู้ใช้สามารถเลือกดูผลการค้นหาร่องรอยที่ไม่ดีประเภท Inappropriate Intimacy (General Form) ได้จากการเลือกเมนู View > Inappropriate Intimacy หรือ Ctrl+I ระบบก็จะทำการแสดงแฟรมที่มีรายละเอียดค่าที่ได้จากการค้นหา แสดงได้ดังรูปที่ ก.9



รูปที่ ก.9 แสดงเฟรมการดูผลการค้นหาร่องรอยที่ไม่ดี ประเภท Inappropriate Intimacy

ก.2.9 การดูผลการค้นหาร่องรอยที่ไม่ดีทั้งหมด

ผู้ใช้สามารถเลือกดูผลการค้นหาร่องรอยที่ไม่ดีประเภททั้งหมด ได้จากการเลือกเมนู View > All หรือ Ctrl+A ระบบก็จะทำการแสดงแฟรมที่มีรายละเอียดค่าที่ได้จากการค้นหา แสดงได้ดังรูปที่ ก.10



รูปที่ ก.10 แสดงเฟรมการดูผลการค้นหาร่องรอยที่ไม่ดีทั้งหมด

ภาคผนวก ข

ซอร์สโค้ดของโปรแกรมที่นำมาทดสอบ

ข.1 โปรแกรมที่ 1

```
import java.util.Date;
import java.util.Enumeration;
import java.util.Vector;

class Movie {
	public static final int CHILDRENS = 2;
	public static final int REGULAR = 0;
	public static final int NEW_RELEASE = 1;
	private String _title;
	private int _priceCode;

	public Movie(String title, int priceCode) {
		_title = title;
		_priceCode = priceCode;
	}

	public int getPriceCode() {
		return _priceCode;
	}

	public void setPriceCode(int arg) {
		_priceCode = arg;
	}

	public String getTitle() {
		return _title;
	};
}

class Rental {
	private Movie _movie;
	private int _daysRented;

	public Rental(Movie movie, int daysRented) {
		_movie = movie;
		_daysRented = daysRented;
	}

	public int getDaysRented() {
		return _daysRented;
	}

	public Movie getMovie() {
		return _movie;
	}
}
```

```
class CrediCard {
	private String _id;
	private String _owner;
	private int _totalAmount = 0;

	public CrediCard(String id, String name) {
		_id = id;
		_owner = name;
	}

	public void setTotalAmount(int amount) {
		_totalAmount += amount;
	}

	public String getID() {
		return _id;
	}

	public String getOwner() {
		return _owner;
	}
}

class Report {
	private Date _date;

	public Report(Date today) {
		_date = today;
	}

	public String generate() {
		Customer _customer = new Customer("");
		String result = "Report : " + _date.toString();
		result += _customer.statement() + "\n";
		return result;
	}
}


class Collections<E>{
	 private Vector ele;
	 public void addElement(E e){
		 ele.add(e);
	 }
	 public Enumeration elements(){
		 return (Enumeration) ele;
	 }
}


class Customer {
	private String _name;
	//private Vector _rentals = new Vector();
	private Collections _rentals = new Collections();
	public Customer(String name) {
		_name = name;
	};
	public void addRental(Rental arg) {
		_rentals.addElement(arg);
	}
	public String getName() {
		 return _name;
	};
```

```
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        Enumeration rentals = _rentals.elements();
        String result = "Rental Record for " + getName() + "\n";
        while (rentals.hasMoreElements()) {
                double thisAmount = 0;
                Rental each = (Rental) rentals.nextElement();
                // determine amounts for each line
                switch (each.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                        thisAmount += 2;
                        if (each.getDaysRented() > 2)
                                thisAmount += (each.getDaysRented() - 2) * 1.5;
                        break;
                case Movie.NEW_RELEASE:
                        thisAmount += each.getDaysRented() * 3;
                        break;
                case Movie.CHILDRENS:
                        thisAmount += 1.5;
                        if (each.getDaysRented() > 3)
                                thisAmount += (each.getDaysRented() - 3) * 1.5;
                        break;
                }
                // add frequent renter points
                frequentRenterPoints++;
                // add bonus for a two day new release rental
                if ((each.getMovie().getPriceCode() == Movie.NEW_RELEASE)
                                && each.getDaysRented() > 1)
                        frequentRenterPoints++;
                // show figures for this rental
                result += "\t" + each.getMovie().getTitle() + "\t"
                                + String.valueOf(thisAmount) + "\n";
                totalAmount += thisAmount;
        }
        // add footer lines
        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints)
                        + " frequent renter points";
        return result;
}
```

ข.2 โปรแกรมที่ 2

```
class ArrayListDemo<E> {
     private int modCount = 0;
     private int size = 1;
     private transient Object[] elementData;
     public void ensureCapacity(int minCapacity) {
        modCount++;
        int oldCapacity = elementData.length;
        if (minCapacity > oldCapacity) {
            Object oldData[] = elementData;
            int newCapacity = (oldCapacity * 3) / 2 + 1;
            if (newCapacity < minCapacity)
                newCapacity = minCapacity;
                elementData = Arrays.copyOf(elementData, newCapacity);
        }
    }
```

```
    public boolean add(E e) {
        ensureCapacity(size + 1);
        elementData[size++] = e;
        return true;
    }
    public int size() {
        return size;
    }
    public E get(int index) {
        rangeCheck(index);
        return (E) elementData(index);
    }
    private void rangeCheck(int index) {
        if (index >= size)
           throw new IndexOutOfBoundsException(outOfBoundsMsg(index));
    }
    String elementData(int index) {
        return (String) elementData[index];
    }
    private String outOfBoundsMsg(int index) {
        return "Index: " + index + ", Size: " + size;
    }
    public E remove(int index) {
        rangeCheck(index);
        modCount++;
        String oldValue = elementData(index);
        int numMoved = size - index - 1;
        if (numMoved > 0)
            System.arraycopy(elementData, index + 1, elementData, index,
                             numMoved);
            elementData[--size] = null;
            return (E) oldValue;
       }
}

class Queue<E> {
    public static final int GRADE = 1;
    ArrayListDemo delegate = new ArrayListDemo();
    public void addRear(E s) {
       delegate.add(s);
    }
    public void addManager(Manager m) {
       delegate.add(m);
    }
    public String removeFront() {
       int i = 0;
       String result = delegate.get(i).toString();
       delegate.remove(i);
       return result;
    }
    public ArrayListDemo getSize() {
       return delegate;
    }
}
```

```
class Report {
    Queue q = new Queue();
    public void generate() {
        String s = "";
        Manager m = new Manager("", "", Queue.GRADE);
        q.addRear(s);
        q.addRear(m);
        q.addManager(m);
        q.getSize().size();
    }
}

class Manager extends Employee {
    private int _grade;
    public Manager(String name, String id, int grade) {
        _name = name;
        _id = id;
        _grade = grade;
        if (isPriviliged())
              assignCar();
    }
    boolean isPriviliged() {
        return _grade > Queue.GRADE;
    }
}

class Employee {
    protected String _name;
    protected String _id;
    boolean isPriviliged() {
        return true;
    }
    void assignCar() {
    }
}
```

ข.3 โปรแกรมที่ 3

```
public class Robot {
    Machine location;
    String bin;

    public Robot() {}

    public Machine location() {return location;}
    public void moveTo(Machine location) {this.location = location;}

    public void pick() {this.bin = location.take();}
    public String bin() {return bin;}

    public void release() {
        location.put(bin);
        bin = null;
    }
}
```

```
public class Report {
    public static void report(Writer out, List machines, Robot robot)
            throws IOException
    {
        out.write("FACTORY REPORT\n");

        Iterator line = machines.iterator();
        while (line.hasNext()) {
            Machine machine = (Machine) line.next();
            out.write("Machine " + machine.name());

            if (machine.bin() != null)
                out.write(" bin=" + machine.bin());

            out.write("\n");
        }
        out.write("\n");

        out.write("Robot");
        if (robot.location() != null)
            out.write(" location=" + robot.location().name());

        if (robot.bin() != null)
            out.write(" bin=" + robot.bin());

        out.write("\n");

        out.write("========\n");
    }
}
public class Machine {
    String name;
    String location;
    String bin;

    public Machine(String name, String location) {
        this.name = name;
        this.location = location;
    }

    public String take() {
        String result = bin;
        bin = null;
        return result;
    }

    public String bin() {
        return bin;
    }

    public void put(String bin) {
        this.bin = bin;
    }

    public String name() {return name;}
}
```

```
class CreateRobot {
     String _name;
     public CreateRobot(String name) {
         _name = name;
     }
     public void testRobot() {
         Machine sorter = new Machine("Sorter", "left");
         sorter.put("chips");
         Machine oven = new Machine("Oven", "middle");
         Robot robot = new Robot();
         robot.moveTo(sorter);
         robot.pick();
         robot.moveTo(oven);
         robot.release();
     }
}

class CreateReport {
     String _name;
     public CreateReport(String name) {
         _name = name;
     }
     public void testReport() throws IOException {
         ArrayList line = new ArrayList();
         line.add(new Machine("mixer", "left"));
         Machine extruder = new Machine("extruder", "center");
         extruder.put("paste");
         line.add(extruder);
         Machine oven = new Machine("oven", "right");
         oven.put("chips");
         line.add(oven);
         Robot robot = new Robot();
         robot.moveTo(extruder);
         robot.pick();
         StringWriter out = new StringWriter();
         Report2.report(out, line, robot);
         String expected = "FACTORY REPORT\n"
              + "Machine mixer\nMachine extruder\n"
              + "Machine oven bin=chips\n\n"
              + "Robot location=extruder bin=paste\n" + "========\n";
       }
}
```

ภาคผนวก ค

ข้อเท็จจริงของโปรแกรมที่นำมาทดสอบ

ค.1 โปรแกรมที่ 1

```
jclass(Movie, , 7, 30);
jclass(Rental, , 32, 48);
jclass(CrediCard, , 50, 71);
jclass(Report, , 73, 86);
jclass(Customer, , 88, 145);
jfield(CHILDRENS, int, 8, 8);
owns(Movie, CHILDRENS);
jfield(REGULAR, int, 9, 9);
owns(Movie, REGULAR);
jfield(NEW_RELEASE, int, 10, 10);
owns(Movie, NEW_RELEASE);
jfield(_title, String, 11, 11);
owns(Movie, _title);
jfield(_priceCode, int, 12, 12);
owns(Movie, _priceCode);
jfield(_movie, Movie, 33, 33);
owns(Rental, _movie);
jfield(_daysRented, int, 34, 34);
owns(Rental, _daysRented);
jfield(_id, String, 51, 51);
owns(CrediCard, _id);
jfield(_owner, String, 52, 52);
owns(CrediCard, _owner);
jfield(_totalAmount, int, 53, 53);
owns(CrediCard, _totalAmount);
jfield(_date, Date, 74, 74);
owns(Report, _date);
jfield(_name, String, 89, 89);
owns(Customer, _name);
jfield(_rentals, Vector, 90, 90);
owns(Customer, _rentals);
jmethod(getPriceCode[], int, null, 19, 21);
owns(Movie, getPriceCode[]);
jmethod(setPriceCode[int], void, [int arg], 23, 25);
owns(Movie, setPriceCode[int]);
jmethod(getTitle[], String, null, 27, 29);
owns(Movie, getTitle[]);
jmethod(getDaysRented[], int, null, 41, 43);
owns(Rental, getDaysRented[]);
jmethod(getMovie[], Movie, null, 45, 47);
owns(Rental, getMovie[]);
jmethod(setTotalAmount[int], void, [int amount], 60, 62);
owns(CrediCard, setTotalAmount[int]);
jmethod(getID[], String, null, 64, 66);
owns(CrediCard, getID[]);
jmethod(getOwner[], String, null, 68, 70);
owns(CrediCard, getOwner[]);
```

```
jmethod(generate[], String, null, 80, 85);
owns(Report, generate[]);
jmethod(addRental[Rental], void, [Rental arg], 96, 98);
owns(Customer, addRental[Rental]);
jmethod(getName[], String, null, 100, 102);
owns(Customer, getName[]);
jmethod(statement[], String, null, 104, 144);
owns(Customer, statement[]);
jexpression(_date.toString[], null, _date, 82);
owns(generate[], _date.toString[]);
owns(Report, _date.toString[]);
jscope(Date, _date.toString[]);
jexpression(_customer.statement[], null, _customer, 83);
owns(generate[], _customer.statement[]);
owns(Report, _customer.statement[]);
jscope(Customer, _customer.statement[]);
jequals(statement[], _customer.statement[]);
jexpression(_rentals.addElement[Rental], [arg], _rentals, 97);
owns(addRental[Rental], _rentals.addElement[Rental]);
owns(Customer, _rentals.addElement[Rental]);
jscope(Vector, _rentals.addElement[Rental]);
jexpression(_rentals.elements[], null, _rentals, 107);
owns(statement[], _rentals.elements[]);
owns(Customer, _rentals.elements[]);
jscope(Vector, _rentals.elements[]);
jexpression(null.getName[], null, null, 108);
owns(statement[], null.getName[]);
owns(Customer, null.getName[]);
jscope(Customer, null.getName[]);
jequals(statement[], null.getName[]);
jexpression(rentals.hasMoreElements[], null, rentals, 109);
owns(statement[], rentals.hasMoreElements[]);
owns(Customer, rentals.hasMoreElements[]);
jscope(Enumeration, rentals.hasMoreElements[]);
jexpression(rentals.nextElement[], null, rentals, 111);
owns(statement[], rentals.nextElement[]);
owns(Customer, rentals.nextElement[]);
jscope(Enumeration, rentals.nextElement[]);
jexpression(each.getMovie[].getPriceCode[], null, each.getMovie[], 113);
owns(statement[], each.getMovie[].getPriceCode[]);
owns(Customer, each.getMovie[].getPriceCode[]);
jscope(Rental, Movie, each.getMovie[].getPriceCode[]);
jexpression(each.getDaysRented[], null, each, 116);
owns(statement[], each.getDaysRented[]);
owns(Customer, each.getDaysRented[]);
jscope(Rental, each.getDaysRented[]);
jequals(getDaysRented[], each.getDaysRented[]);
jexpression(each.getDaysRented[], null, each, 117);
jexpression(each.getDaysRented[], null, each, 120);
jexpression(each.getDaysRented[], null, each, 124);
jexpression(each.getDaysRented[], null, each, 125);
jexpression(each.getMovie[].getPriceCode[], null, each.getMovie[], 131);
jexpression(each.getDaysRented[], null, each, 132);
jexpression(each.getMovie[].getTitle[], null, each.getMovie[], 135);
owns(statement[], each.getMovie[].getTitle[]);
owns(Customer, each.getMovie[].getTitle[]);
jscope(Rental, Movie, each.getMovie[].getTitle[]);
jexpression(String.valueOf[double], [thisAmount], String, 136);
owns(statement[], String.valueOf[double]);
```

```
owns(Customer, String.valueOf[double]);
jexpression(String.valueOf[double], [totalAmount], String, 140);
jexpression(String.valueOf[int], [frequentRenterPoints], String, 141);
owns(statement[], String.valueOf[int]);
owns(Customer, String.valueOf[int]);
jfieldaccess(Movie.REGULAR, Movie, 114, 114);
jscope(Movie, Movie.REGULAR);
owns(Customer, Movie.REGULAR);
jequals(REGULAR, Movie.REGULAR);
jfieldaccess(Movie.NEW_RELEASE, Movie, 119, 119);
jscope(Movie, Movie.NEW_RELEASE);
owns(Customer, Movie.NEW_RELEASE);
jequals(NEW_RELEASE, Movie.NEW_RELEASE);
jfieldaccess(Movie.CHILDRENS, Movie, 122, 122);
jscope(Movie, Movie.CHILDRENS);
owns(Customer, Movie.CHILDRENS);
jequals(CHILDRENS, Movie.CHILDRENS);
jfieldaccess(Movie.NEW_RELEASE, Movie, 131, 131);
jconstructor(Movie, [String title, int priceCode], 14, 17);
owns(Movie, Movie);
jconstructor(Rental, [Movie movie, int daysRented], 36, 39);
owns(Rental, Rental);
jconstructor(CrediCard, [String id, String name], 55, 58);
owns(CrediCard, CrediCard);
jconstructor(Report, [Date today], 76, 78);
owns(Report, Report);
jconstructor(Customer, [String name], 92, 94);
owns(Customer, Customer);
jattribute(arg, int, 23, 23);
owns(setPriceCode[int], arg);
jattribute(amount, int, 60, 60);
owns(setTotalAmount[int], amount);
jattribute(_customer, Customer, 81, 81);
owns(generate[], _customer);
jattribute(result, String, 82, 82);
owns(generate[], result);
jattribute(arg, Rental, 96, 96);
owns(addRental[Rental], arg);
jattribute(totalAmount, double, 105, 105);
owns(statement[], totalAmount);
jattribute(frequentRenterPoints, int, 106, 106);
owns(statement[], frequentRenterPoints);
jattribute(rentals, Enumeration, 107, 107);
owns(statement[], rentals);
jattribute(result, String, 108, 108);
owns(statement[], result);
jattribute(thisAmount, double, 110, 110);
owns(statement[], thisAmount);
jattribute(each, Rental, 111, 111);
owns(statement[], each);
```

ค.2 โปรแกรมที่ 2

```
jclass(ArrayListDemo, , 5, 61);
jclass(Queue, , 63, 85);
jclass(Report, , 87, 98);
jclass(Manager, , 100, 114);
jclass(Employee, , 116, 126);
jfield(modCount, int, 6, 6);
owns(ArrayListDemo, modCount);
jfield(size, int, 7, 7);
```

```
owns(ArrayListDemo, size);
jfield(elementData, Object[], 8, 8);
owns(ArrayListDemo, elementData);
jfield(GRADE, int, 64, 64);
owns(Queue, GRADE);
jfield(delegate, ArrayListDemo, 65, 65);
owns(Queue, delegate);
jfield(q, Queue, 88, 88);
owns(Report, q);
jfield(_grade, int, 101, 101);
owns(Manager, _grade);
jfield(_name, String, 117, 117);
owns(Employee, _name);
jfield(_id, String, 118, 118);
owns(Employee, _id);
jmethod(ensureCapacity[int], void, [int minCapacity], 10, 20);
owns(ArrayListDemo, ensureCapacity[int]);
jmethod(add[E], boolean, [E e], 22, 26);
owns(ArrayListDemo, add[E]);
jmethod(size[], int, null, 28, 30);
owns(ArrayListDemo, size[]);
jmethod(get[int], E, [int index], 32, 35);
owns(ArrayListDemo, get[int]);
jmethod(rangeCheck[int], void, [int index], 37, 40);
owns(ArrayListDemo, rangeCheck[int]);
jmethod(elementData[int], String, [int index], 42, 44);
owns(ArrayListDemo, elementData[int]);
jmethod(outOfBoundsMsg[int], String, [int index], 46, 48);
owns(ArrayListDemo, outOfBoundsMsg[int]);
jmethod(remove[int], E, [int index], 50, 60);
owns(ArrayListDemo, remove[int]);
jmethod(addRear[E], void, [E s], 67, 69);
owns(Queue, addRear[E]);
jmethod(addManager[Manager], void, [Manager m], 71, 73);
owns(Queue, addManager[Manager]);
jmethod(removeFront[], String, null, 75, 80);
owns(Queue, removeFront[]);
jmethod(getSize[], ArrayListDemo, null, 82, 84);
owns(Queue, getSize[]);
jmethod(generate[], void, null, 90, 97);
owns(Report, generate[]);
jmethod(isPriviliged[], boolean, null, 111, 113);
owns(Manager, isPriviliged[]);
jmethod(isPriviliged[], boolean, null, 120, 122);
owns(Employee, isPriviliged[]);
jmethod(assignCar[], void, null, 124, 125);
owns(Employee, assignCar[]);
jexpression(Arrays.copyOf[Object[], int], [elementData, newCapacity],
owns(ensureCapacity[int], Arrays.copyOf[Object[], int]);
owns(ArrayListDemo, Arrays.copyOf[Object[], int]);
jexpression(null.ensureCapacity[], [size + 1], null, 23);
owns(add[E], null.ensureCapacity[]);
owns(ArrayListDemo, null.ensureCapacity[]);
jscope(ArrayListDemo, null.ensureCapacity[]);
jequals(add[E], null.ensureCapacity[]);
jexpression(null.rangeCheck[int], [index], null, 33);
owns(get[int], null.rangeCheck[int]);
owns(ArrayListDemo, null.rangeCheck[int]);
jscope(ArrayListDemo, null.rangeCheck[int]);
jequals(get[int], null.rangeCheck[int]);
jexpression(null.elementData[int], [index], null, 34);
owns(get[int], null.elementData[int]);
owns(ArrayListDemo, null.elementData[int]);
jscope(ArrayListDemo, null.elementData[int]);
jequals(get[int], null.elementData[int]);
```

```
jexpression(null.outOfBoundsMsg[int], [index], null, 39);
owns(rangeCheck[int], null.outOfBoundsMsg[int]);
owns(ArrayListDemo, null.outOfBoundsMsg[int]);
jscope(ArrayListDemo, null.outOfBoundsMsg[int]);
jequals(rangeCheck[int], null.outOfBoundsMsg[int]);
jexpression(null.rangeCheck[int], [index], null, 51);
owns(remove[int], null.rangeCheck[int]);
jequals(remove[int], null.rangeCheck[int]);
jexpression(null.elementData[int], [index], null, 53);
owns(remove[int], null.elementData[int]);
jequals(remove[int], null.elementData[int]);
jexpression(System.arraycopy[Object[], Object[], int, int], [elementData,
owns(remove[int], System.arraycopy[Object[], Object[], int, int]);
owns(ArrayListDemo, System.arraycopy[Object[], Object[], int, int]);
jexpression(delegate.add[E], [s], delegate, 68);
owns(addRear[E], delegate.add[E]);
owns(Queue, delegate.add[E]);
jscope(ArrayListDemo, delegate.add[E]);
jequals(add[E], delegate.add[E]);
jexpression(delegate.add[Manager], [m], delegate, 72);
owns(addManager[Manager], delegate.add[Manager]);
owns(Queue, delegate.add[Manager]);
jscope(ArrayListDemo, delegate.add[Manager]);
jequals(add[E], delegate.add[Manager]);
jexpression(delegate.get[int].toString[], null, delegate.get[int], 77);
owns(removeFront[], delegate.get[int].toString[]);
owns(Queue, delegate.get[int].toString[]);
jscope(ArrayListDemo, E, delegate.get[int].toString[]);
jexpression(delegate.remove[int], [i], delegate, 78);
owns(removeFront[], delegate.remove[int]);
owns(Queue, delegate.remove[int]);
jscope(ArrayListDemo, delegate.remove[int]);
jequals(remove[int], delegate.remove[int]);
jexpression(q.addRear[String], [s], q, 93);
owns(generate[], q.addRear[String]);
owns(Report, q.addRear[String]);
jscope(Queue, q.addRear[String]);
jequals(addRear[E], q.addRear[String]);
jexpression(q.addRear[Manager], [m], q, 94);
owns(generate[], q.addRear[Manager]);
owns(Report, q.addRear[Manager]);
jscope(Queue, q.addRear[Manager]);
jequals(addRear[E], q.addRear[Manager]);
jexpression(q.addManager[Manager], [m], q, 95);
owns(generate[], q.addManager[Manager]);
owns(Report, q.addManager[Manager]);
jscope(Queue, q.addManager[Manager]);
jequals(addManager[Manager], q.addManager[Manager]);
jexpression(q.getSize[].size[], null, q.getSize[], 96);
owns(generate[], q.getSize[].size[]);
owns(Report, q.getSize[].size[]);
jscope(Queue, ArrayListDemo, q.getSize[].size[]);
jfieldaccess(elementData.length, elementData, 12, 12);
jscope(elementData, elementData.length);
owns(ArrayListDemo, elementData.length);
jfieldaccess(Queue.GRADE, Queue, 92, 92);
jscope(Queue, Queue.GRADE);
owns(Report, Queue.GRADE);
jequals(GRADE, Queue.GRADE);
jfieldaccess(Queue.GRADE, Queue, 112, 112);
jscope(Queue, Queue.GRADE);
owns(Manager, Queue.GRADE);
jequals(GRADE, Queue.GRADE);
jconstructor(Manager, [String name, String id, int grade], 103, 109);
```

```
owns(Manager, Manager);
jattribute(minCapacity, int, 10, 10);
owns(ensureCapacity[int], minCapacity);
jattribute(oldCapacity, int, 12, 12);
owns(ensureCapacity[int], oldCapacity);
jattribute(oldData[], Object, 14, 14);
owns(ensureCapacity[int], oldData[]);
jattribute(newCapacity, int, 15, 15);
owns(ensureCapacity[int], newCapacity);
jattribute(e, E, 22, 22);
owns(add[E], e);
jattribute(index, int, 32, 32);
owns(get[int], index);
jattribute(index, int, 37, 37);
owns(rangeCheck[int], index);
jattribute(index, int, 42, 42);
owns(elementData[int], index);
jattribute(index, int, 46, 46);
owns(outOfBoundsMsg[int], index);
jattribute(index, int, 50, 50);
owns(remove[int], index);
jattribute(oldValue, String, 53, 53);
owns(remove[int], oldValue);
jattribute(numMoved, int, 54, 54);
owns(remove[int], numMoved);
jattribute(s, E, 67, 67);
owns(addRear[E], s);
jattribute(m, Manager, 71, 71);
owns(addManager[Manager], m);
jattribute(i, int, 76, 76);
owns(removeFront[], i);
jattribute(result, String, 77, 77);
owns(removeFront[], result);
jattribute(s, String, 91, 91);
owns(generate[], s);
jattribute(m, Manager, 92, 92);
owns(generate[], m);
```

ค.3 โปรแกรมที่ 3

```
jclass(Report2, Programe3.java, 10, 31);
jclass(Machine, Programe3.java, 33, 60);
jclass(Robot, Programe3.java, 62, 89);
jclass(CreateRobot, Programe3.java, 91, 109);
jclass(CreateReport, Programe3.java, 111, 137);
jfield(name, String, 34, 34);
owns(Machine, name);
jfield(location, String, 35, 35);
owns(Machine, location);
jfield(bin, String, 36, 36);
owns(Machine, bin);
jfield(location, Machine, 63, 63);
owns(Robot, location);
jfield(bin, String, 64, 64);
owns(Robot, bin);
jfield(_name, String, 92, 92);
owns(CreateRobot, _name);
jfield(_name, String, 112, 112);
owns(CreateReport, _name);
jmethod(report[Writer, List, Robot], void, [Writer out,
       List machines, Robot robot], 11, 30);
```

```
owns(Report2, report[Writer, List, Robot]);
jmethod(take[], String, null, 43, 47);
owns(Machine, take[]);
jmethod(bin[], String, null, 49, 51);
owns(Machine, bin[]);
jmethod(put[String], void, [String bin], 53, 55);
owns(Machine, put[String]);
jmethod(name[], String, null, 57, 59);
owns(Machine, name[]);
jmethod(location[], Machine, null, 69, 71);
owns(Robot, location[]);
jmethod(moveTo[Machine], void, [Machine location], 73, 75);
owns(Robot, moveTo[Machine]);
jmethod(pick[], void, null, 77, 79);
owns(Robot, pick[]);
jmethod(bin[], String, null, 81, 83);
owns(Robot, bin[]);
jmethod(release[], void, null, 85, 88);
owns(Robot, release[]);
jmethod(testRobot[], void, null, 98, 108);
owns(CreateRobot, testRobot[]);
jmethod(testReport[], void, null, 118, 136);
owns(CreateReport, testReport[]);
jexpression(out.write[], ["FACTORY REPORT\n"], out, 13);
owns(report[Writer, List, Robot], out.write[]);
owns(Report2, out.write[]);
jscope(Writer, out.write[]);
jexpression(machines.iterator[], null, machines, 14);
owns(report[Writer, List, Robot], machines.iterator[]);
owns(Report2, machines.iterator[]);
jscope(List, machines.iterator[]);
jexpression(line.hasNext[], null, line, 15);
owns(report[Writer, List, Robot], line.hasNext[]);
owns(Report2, line.hasNext[]);
jscope(Iterator, line.hasNext[]);
jexpression(line.next[], null, line, 16);
owns(report[Writer, List, Robot], line.next[]);
owns(Report2, line.next[]);
jscope(Iterator, line.next[]);
jexpression(out.write[], ["Machine " + machine.name()], out, 17);
jexpression(machine.bin[], null, machine, 18);
owns(report[Writer, List, Robot], machine.bin[]);
owns(Report2, machine.bin[]);
jscope(Machine, machine.bin[]);
jequals(bin[], machine.bin[]);
jexpression(out.write[], [" bin=" + machine.bin()], out, 19);
jexpression(out.write[], ["\n"], out, 20);
jexpression(out.write[], ["\n"], out, 22);
jexpression(out.write[], ["Robot"], out, 23);
jexpression(robot.location[], null, robot, 24);
owns(report[Writer, List, Robot], robot.location[]);
owns(Report2, robot.location[]);
jscope(Robot, robot.location[]);
jequals(location[], robot.location[]);
jexpression(out.write[], [" location=" + robot.location()
        .name()], out, 25);
jexpression(robot.bin[], null, robot, 26);
owns(report[Writer, List, Robot], robot.bin[]);
owns(Report2, robot.bin[]);
jscope(Robot, robot.bin[]);
jequals(bin[], robot.bin[]);
jexpression(out.write[], [" bin=" + robot.bin()], out, 27);
jexpression(out.write[], ["\n"], out, 28);
jexpression(out.write[], ["========\n"], out, 29);
jexpression(location.take[], null, location, 78);
```

```
owns(pick[], location.take[]);
owns(Robot, location.take[]);
jscope(Machine, location.take[]);
jequals(take[], location.take[]);
jexpression(location.put[String], [bin], location, 86);
owns(release[], location.put[String]);
owns(Robot, location.put[String]);
jscope(Machine, location.put[String]);
jequals(put[String], location.put[String]);
jexpression(sorter.put[], ["chips"], sorter, 100);
owns(testRobot[], sorter.put[]);
owns(CreateRobot, sorter.put[]);
jscope(Machine, sorter.put[]);
jequals(put[String], sorter.put[]);
jexpression(robot.moveTo[Machine], [sorter], robot, 103);
owns(testRobot[], robot.moveTo[Machine]);
owns(CreateRobot, robot.moveTo[Machine]);
jscope(Robot, robot.moveTo[Machine]);
jequals(moveTo[Machine], robot.moveTo[Machine]);
jexpression(robot.pick[], null, robot, 104);
owns(testRobot[], robot.pick[]);
owns(CreateRobot, robot.pick[]);
jscope(Robot, robot.pick[]);
jequals(pick[], robot.pick[]);
jexpression(robot.moveTo[Machine], [oven], robot, 105);
jexpression(robot.release[], null, robot, 106);
owns(testRobot[], robot.release[]);
owns(CreateRobot, robot.release[]);
jscope(Robot, robot.release[]);
jequals(release[], robot.release[]);
jexpression(line.add[], [new Machine("mixer", "left")], line, 120);
owns(testReport[], line.add[]);
owns(CreateReport, line.add[]);
jscope(ArrayList, line.add[]);
jexpression(extruder.put[], ["paste"], extruder, 122);
owns(testReport[], extruder.put[]);
owns(CreateReport, extruder.put[]);
jscope(Machine, extruder.put[]);
jequals(put[String], extruder.put[]);
jexpression(line.add[Machine], [extruder], line, 123);
owns(testReport[], line.add[Machine]);
owns(CreateReport, line.add[Machine]);
jscope(ArrayList, line.add[Machine]);
jexpression(oven.put[], ["chips"], oven, 125);
owns(testReport[], oven.put[]);
owns(CreateReport, oven.put[]);
jscope(Machine, oven.put[]);
jequals(put[String], oven.put[]);
jexpression(line.add[Machine], [oven], line, 126);
jexpression(robot.moveTo[Machine], [extruder], robot, 128);
owns(testReport[], robot.moveTo[Machine]);
owns(CreateReport, robot.moveTo[Machine]);
jexpression(robot.pick[], null, robot, 129);
owns(testReport[], robot.pick[]);
owns(CreateReport, robot.pick[]);
jexpression(Report2.report[StringWriter, ArrayList, Robot], [out,
      line, robot], Report2, 131);
owns(testReport[], Report2.report[StringWriter, ArrayList, Robot]);
owns(CreateReport, Report2.report[StringWriter, ArrayList, Robot]);
jfieldaccess(this.bin, this, 54, 54);
```

```
jscope(this, this.bin);
owns(Machine, this.bin);
jfieldaccess(this.location, this, 74, 74);
jscope(this, this.location);
owns(Robot, this.location);
jfieldaccess(this.bin, this, 78, 78);
jscope(this, this.bin);
owns(Robot, this.bin);
jconstructor(Machine, [String name, String location], 38, 41);
owns(Machine, Machine);
jconstructor(Robot, null, 66, 67);
owns(Robot, Robot);
jconstructor(CreateRobot, [String name], 94, 96);
owns(CreateRobot, CreateRobot);
jconstructor(CreateReport, [String name], 114, 116);
owns(CreateReport, CreateReport);
jattribute(out, Writer, 11, 11);
owns(report[Writer, List, Robot], out);
jattribute(machines, List, 11, 11);
owns(report[Writer, List, Robot], machines);
jattribute(robot, Robot, 11, 11);
owns(report[Writer, List, Robot], robot);
jattribute(line, Iterator, 14, 14);
owns(report[Writer, List, Robot], line);
jattribute(machine, Machine, 16, 16);
owns(report[Writer, List, Robot], machine);
jattribute(result, String, 44, 44);
owns(take[], result);
jattribute(bin, String, 53, 53);
owns(put[String], bin);
jattribute(location, Machine, 73, 73);
owns(moveTo[Machine], location);
jattribute(sorter, Machine, 99, 99);
owns(testRobot[], sorter);
jattribute(oven, Machine, 101, 101);
owns(testRobot[], oven);
jattribute(robot, Robot, 102, 102);
owns(testRobot[], robot);
jattribute(line, ArrayList, 119, 119);
owns(testReport[], line);
jattribute(extruder, Machine, 121, 121);
owns(testReport[], extruder);
jattribute(oven, Machine, 124, 124);
owns(testReport[], oven);
jattribute(robot, Robot, 127, 127);
owns(testReport[], robot);
jattribute(out, StringWriter, 130, 130);
owns(testReport[], out);
jattribute(expected, String, 132, 135);
owns(testReport[], expected);
```

ประวัติผู้เขียนวิทยานิพนธ์

นายขันติ ยี่สุ่น เกิดเมื่อวันที่ 12 มกราคม พ.ศ. 2524 สำเร็จการศึกษาหลักสูตรวิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิทยาการคอมพิวเตอร์ ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยบูรพา เมื่อปีการศึกษา 2547 และ เข้าศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ปีการศึกษา 2548