NTNU – Trondheim
Norwegian University of
Science and Technology

Department of Computer and Information Sciences

Examination paper for TDT 4242 Software Requirements and Testing

Academic contact during examination: Carl-Fredrik Sørensen

Phone: 951 19 690

Examination date: Mai 16th, 2015

Examination time: 09:00 AM to 1:00 PM

Permitted examination support material: Code A – All printed and handwritten material is allowed, pocket calculators allowed

Other information: Exam developed by Carl-Fredrik Sørensen and checked by Professor Tor Stålhane

Language: English

Number of pages: 15

Number of pages enclosed: 6

Checked by:

Date Signature

Introduction

In this exam, you can score a maximum of 50 points. The remaining 50 points for the semester comes from the compulsory exercises.

If you feel that any of the problems require information that you do not find in the text, then you should

- Document the necessary assumptions
- Explain why you need them

Your answers should be brief and to the point.

Problem 1 – Multiple choice (10 points)

Use the two attached answer forms for this task (keep one for yourself). You can get a new form from the invigilator if necessary. Only one answer is completely correct. For every question, a correct answer counts as 1 point. An incorrect answer or more than one answer counts as -0.5 points. A question left blank counts as 0 points. The lowest score is 0 point on this task.

1) **Which factors is the most challenging with respect to the 'Software Crisis'?**
   a) The development environment
   b) The lack of customer involvement
   c) Imprecise and incomplete requirements
   d) Bad testing

2) **What is the main reason for software failures?**
   a) Missing and bad testing
   b) The lack of understanding of what to develop
   c) Reuse of software components
   d) The lack of traceability from requirements to finished product

3) **Which quality metric for requirements is used to calculate if a requirement is part of the domain to be developed for?**
   a) Opacity
   b) Ambiguity
   c) Noise
   d) Forward referencing

4) **What are NOT goal-oriented requirement activities**
   a) Modelling goals
   b) Verifying goals
   c) Identifying goals
   d) Implementing goals

5) **What is the most relevant issue when deciding on delivering a software version to customers**
   a) Amount of accomplished tested and amount of found bugs
   b) Amount of found bugs, probability that they will occurs and their impacts
   c) Amount of bugs and necessary effort to correct them
   d) Number of stakeholders and their requirements

6) **A textual use case mainly describes …**
   a) **All steps in an interaction between an actor and the system to fulfill a specific functional goal, including exceptions**
   b) All steps the system need to do to fulfill a functional goal, including exceptions
   c) All steps the user need to do to fulfill a functional goal, including exceptions
   d) The objective of the system

7) **In the maintenance phase, the product needs to be tested against earlier test cases. This is also known as __________testing.**
   a) Unit
   b) Integration
   c) **Regression**
   d) Domain

8) **A disadvantage with textual use cases is that**
   a) They are harder to understand than complex use case diagrams
   b) They are difficult to use for further system modelling, e.g. interaction diagrams
   c) They do not specify action sequences
   d) **They require a lot of work to develop**

9) **A mis-use case is most useful for**
   a) Developing tests for the whole value domain
   b) Identifying actions that can harm the system under normal use
   c) **Identifying threats and new requirements for the system**
   d) Specifying how the user interface should be designed

10) **The main problem with domain testing is …**
   a) To identify the correct predicates
   b) To get through all path-conditions
   c) Precision on input values
   d) **The number of tests required if many predicates**

11) **Which best captures the nature of the quality paradigm?**
   a) **The Nature of Quality, A Process Perspective, Defect Elimination**
   b) Measurement, Quality Control, Validation
   c) Feasibility, Requirements, Economics, Customer's Needs
   d) Analysis, Testing, Design

12) **Which of these are gray box test technique?**
   a) Code coverage testing
   b) **Path testing**
   c) Domain testing
   d) Mutation testing

13) **Which of the sentences have open boundaries in path domains`?**
   a) IF (A == 45 AND B >= 10) THEN …
   b) IF (A > 30 OR B == 10) THEN …
   c) IF (A >= 10 AND B < 10.0) THEN …
   d) **IF (A > 0 AND B < 10) THEN**

14) **What is the main advantage with test-driven development?**
   a) **Simpler debugging**
   b) Test strategy is chosen explicitly
   c) The requirements do not need to be very explicit
   d) The customer are kept informed about the development progression

15) How many tests is necessary to make to get complete test coverage if you have five AND-predicates in an IF-statement?

a) 5
b) 16
c) 32
d) 50

16) A test double is often suitable to

a) Avoid that testing takes too much time
b) Avoid that unclear test requirements are specified
c) Be able to develop and test components in isolation
d) All are correct

17) A software engineer is measuring the quality of a software system. She is concerned with the "reliability" and the "validity" of her measurements. Which of the following is true?

a) Reliability refers to the extent to which the measurement represents the actual quality of the system and validity refers to the consistency of her quality measurements
b) Reliability refers to the consistency of her quality measurements and validity refers to the extent to which the measurement represents the actual quality of the system
c) Reliability refers to the accuracy of her quality measurements and validity refers to the extent to which the measurement follows a quality standard
d) Reliability refers to the concurrency of her quality measurements and validity refers to the extent to which the measurements are consistent with established norms

18) Which of the following is NOT a test double

a) Assertion
b) Fake
c) Mock
d) Stub

19) INVEST i gode user stories. Kravene bør være

a) Irreplaceable, Noteworthy, Valuable, Estimatable, Small, Testable
b) Irreplaceable, Negotiable, Valuable, Estimateable, Stringent, Testable
c) Independent, Noteworthy, Valueable, Estimateable, Stringent, Testable
d) Independent, Negotiable, Valueable, Estimateable, Small, Testable

20) Which method is not suitable for integrating security issues in software requirement?

a) Misuse case
b) i* framework
c) Class diagram
d) Sequence diagram

HAND-IN

Answer form: Task 1

Taskno	A	B	C	D
1.1			X	
1.2		X		
1.3			X	
1.4				X
1.5		X		
1.6	X			
1.7			X	
1.8				X
1.9			X	
1.10				X
1.11	X			
1.12		X		
1.13				X
1.14	X			
1.15			X	
1.16			X	
1.17		X		
1.18	X			
1.19				X
1.20				X

Problem 2 – Requirements engineering (20 points)

2a – Requirement specification – 15 points

1. Based on the description in Appendix 1, specify requirements for environmental monitoring of the food production company. You can yourself decide how to such requirements should be specified. Justify your choice of method.

Three main functions are temperature control, air quality control and water quality control. Correspondingly, functions are measurement, threshold detection and shelf life calculation and alarm notification

Food traceability

- The system should be able to trace all the food

Measurement

- The system should be able to measure the temperature
- The system should be able to measure the time
- The system should be able to measure the humidity

Threshold detection

- The system should be able to detect the threshold combination of temperature and time for cooling chain. i.e. higher than 4°C more than 10 minutes
- The system should be able to detect the threshold combination of temperature and time for cooling chain, i.e. when cooling down a product, the core temperature must be the same as the surface- and target temperature within 30 minutes.
- The system should be able to detect the threshold combination of temperature and time for heating function. E.g., higher than 70°C longer than 10 minutes.

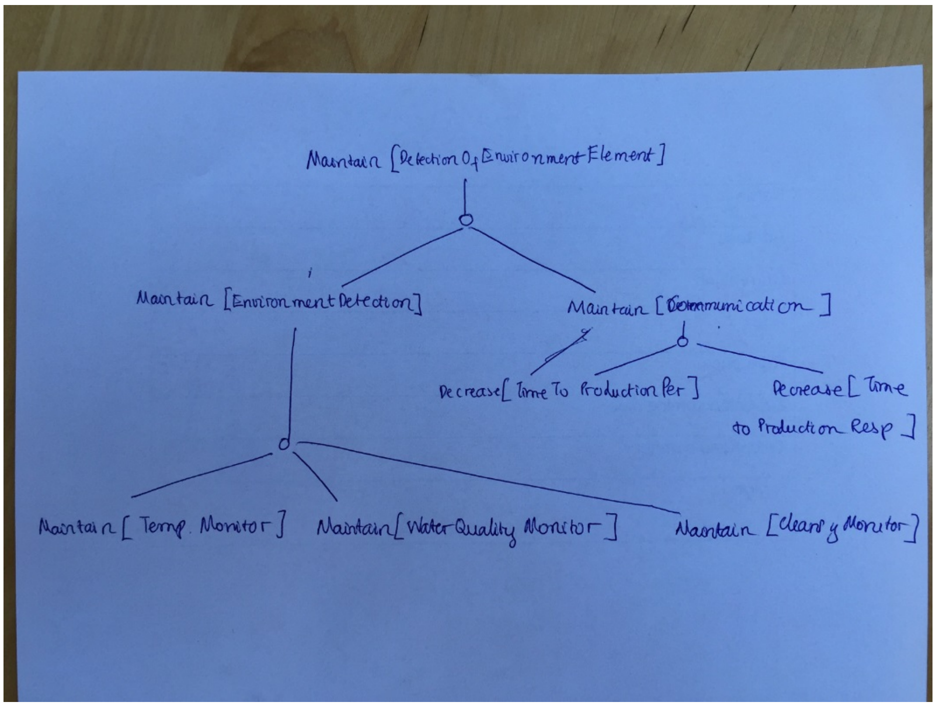
Shelf life calculation

- The system should be able to store measured information over time
- The system should be able to calculate the shelf life time

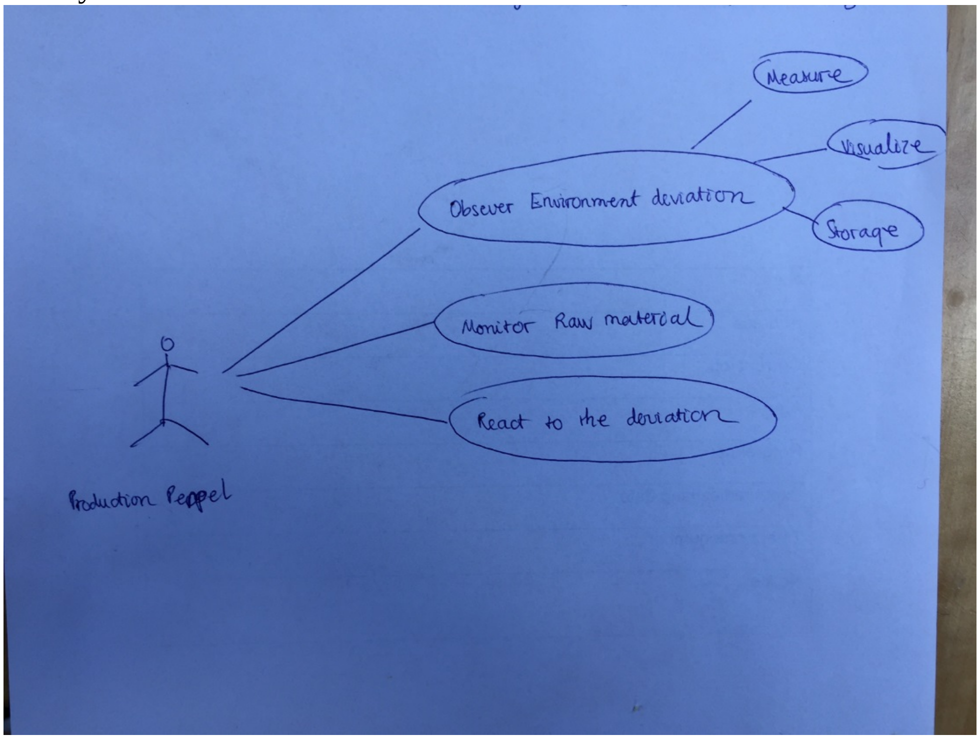
Alarm notification

- The system should be able to decide about the relationships between temperatures and time,
- The system should be able to decide about the relationships between water and air quality,
- The system should be able to decide about the deviation from cleaning procedures

2. Use the defined set of temporal patterns – see appendix 2 – to describe the three upper levels of the requirements to the alarm unit in the production locality.


Maintain [Detection Of Environment Element]
Maintain [Environment Detection]
Maintain [Communication]
Decrease [Time To Production Per]
Decrease [Time to Production Resp]
Maintain [Temp. Monitor]
Maintain [Water Quality Monitor]
Maintain [Cleaning Monitor]

3. Make mis-use cases to the new system of the food production company, describe and explain what these would mean in the development of the system for monitoring the production and storage environments.
4. Make a use case diagram of the most important functionality and operations in the new system.



5. Specify the five most important non-functional requirements the system needs to fulfil.
   - Safety: the system should not cause and injury or damage for users
   - Reliability: the system should correctly measure temperature and time in 99% of operation time
   - Maintainability: the damaged sensors should be replaced within 2 hours

- Tracebility: all information about sensors, temperature, time and food should be storage and queriable within 10 minutes.
- Performance: the system should work properly when running 24 hours continuously.

2b – Textual use cases – 5 points

1. Specify the three most important textual use cases based on your requirements from Problem 2a. Justify why you mean these three are the most important.

Problem 3 – Testing methods (10 points)

It is very important for the food producer that their traceability system is well tested.

1. Specify a test strategy for implementation of the system in the food production company. Explain why this is a good test strategy.
    a. Stakeholder should be manufaturer employee (actual end users), testers, project owner, 3rd party provider (sensor provider)
    b. As the system composes of many different elements, the test should be done in unit, system and integration level
        i. Unit level: black box testing
        ii. Integration level: combination of components
        iii. System level: scenario testing, full path covergage, acceptance testing
    c. At first this is a complex system. So it is good to use scenario testing to connect the product to the described requirements
    d. Blackbox testing is the suitable approach
    e. Acceptance testing and reliability testing is important to decide the completion of the testing.

2. Select one or more test methods to test out the traceability system, from the receipt of raw materials to shipped end product. Explain in every case why you would choose this method.
    a. The system is complex and need the coordination of many components, therefore, dynamic testing is suitable. Some test cases are described as in the Section 3.

3. Select a test method for the environmental monitoring, and write four complete test given the method you chose.
    a. Scenario testing.
    b. Test scenario 1
        i. Product A is delivered to the company, then unpacked and stored at -10 C degree and later shipped to other stored
    c. Test scenario 2
        i. Product B is delivered to the company, then unpacked and stored at -10 C degree and later sent to the production department, transforming by cooling at the degree of -20 C degree
    d. Test scenario 3

        i. Product C is delivered to the company, then unpacked and stored at -10 C degree and later sent to the production department, transforming by heating at the degree of of 100 C degree
    e. Test scenario 4
        i. Product A, B and C are moved from storage to transformation and combination to product D

Problem 4 – Inspection (10 points)

Appendix 3 shows Java code which it is suspected to contain defects.
Inspect the code and identify any issues or defects that you believe is in there, and eventually suggest improvements. Use the following setup to show the results of the inspection, preferably use a table form:

Line number - type of issue - identified issue - suggested improvement

Appendix 4 contains a checklist for inspection of Java.

14. Extra statement, array messages is never used
22. Extra statement, don’t need the variable length
24. Wrong statement, “Good bye” instead of “Bye”
3. Wrong statement, line 19, Boolean listening should have been true
42. Missing statement, line 76, no exception handling
43. Missing statement, line 75, nothing would be written to the file unless it is closed
41. Wrong statement, line 73, for appending the constructor parameter should have been true, not false
40. Extra statement, do not need line 72

EncryptionServerThread.java

```
// separate thread for each client
class EncryptionServerThread extends Thread {
// connection to the client
private Socket socket = null;
// constructor
public EncryptionServerThread(Socket socket) {
        this.socket = socket;
}
// keep talking to the client, until the SecretEncryptor sends "Bye" to us
public void run() {
        System.out.println("Got a new client");
        try {
                PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
                BufferedReader in = new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
                String inputLine, outputLine;
                out.println("Welcome to the encryption server");
                while ((inputLine = in.readLine()) != null) {
                outputLine = "output>>" + SecretEncryptor.encrypt(inputLine);
                out.println(outputLine);
```

```
                        if (outputLine.equals("Bye")) break;
            }
            out.close();
            in.close();
            socket.close();
} catch (IOException e) {
            e.printStackTrace();
            }
            }
}
```

Appendix 1 – Food traceability

The figure below shows a general state diagram of food production in a food production company situated at a single location. The company receives raw materials from other companies, treats, mixes and further refines these through different production processes before the food items are packed for shipping to food stores and restaurants. The company wants to increase the food safety, thus they want establish a hardware and software infrastructure, which can monitor the production and storage environment. Threats towards food safety is e.g. contamination (food poisoning) in incoming raw materials, bad heating- or cooling/freezing treatments internally in the company, bad cleaning, and use of raw materials that have passed their expiration date or shelf life (e.g., exposure to higher storage temperatures than needed to keep the product fresh).

To improve the food traceability in case of an event (outbreak of a disease connected to food consumption), it is a wish to collect information about the environmental state when receiving, producing, packing and shipping products. Data collection can happen in all transitions between states in addition to data that can be collected while in a certain state (e.g, storage temperature). The company has acquired modern traceability equipment and sensors like RFID and other electronic tagging which can be used to identify all traceable units in the whole production plant. These can be automatically read when going from one state to another (e.g. out of storage, in to storage). The company has in addition acquired electronic monitoring of temperatures of water, air and within the production equipment, air quality (humidity, gases (e.g. ammoniac) and dust content)), and water quality. The collected data can be used for different purposes:

1) Calculate remaining shelf life (expiration date)
2) Calculate the consumption of raw materials to minimize rate of loss
3) Establish traceability chain from an item is received into stock, through production and refining, to transport out of the production plant.
4) Establish a service where a traceability number for a certain product can be used to find the production history of the product, including environmental data, and which raw materials that were used in the production of it.
5) Trace the usage of a certain raw material in the production and which end products that contain this material. Note that all raw materials also are tagged with information and a traceability number for each traceable unit (e.g., a sack of spices).

Some rules of thumb to calculate remaining shelf life are:

1) Products that are in a cooling chain cannot be exposed to higher temperatures over a certain amount of time to reduce the remaining shelf life. E.g. higher than 4°C more than 10 minutes
2) Products that are to be heated, need a heating temperature above a given temperature for a certain amount of time. E.g., higher than 70°C longer than 10 minutes.
3) When cooling down a product, the core temperature must be the same as the surface- and target temperature within 30 minutes.
4) Long exposure to sunlight and/or temperatures between 5°C-40°C may cause a very high bacterial growth and make the food item inedible or even poisonous.

The company wishes to establish alarms connected to the storage and production to discover deviations in temperatures, water quality, air quality, and cleaning procedures, in addition to spending of out of shelf life products from the storage into the production. The alarm is to be communicated to both the production personnel as well as the production responsible if a possible deviation is under development or has already happened. The alarm will have three states: Grønn – No deviation, yellow – a possible deviation has been registered, and red – a deviation has been registered. The alarm state is decided by the relationships between temperatures and time, and water and air quality, and deviation from cleaning procedures.

Every single raw material will have some attributes registered or attached with it:

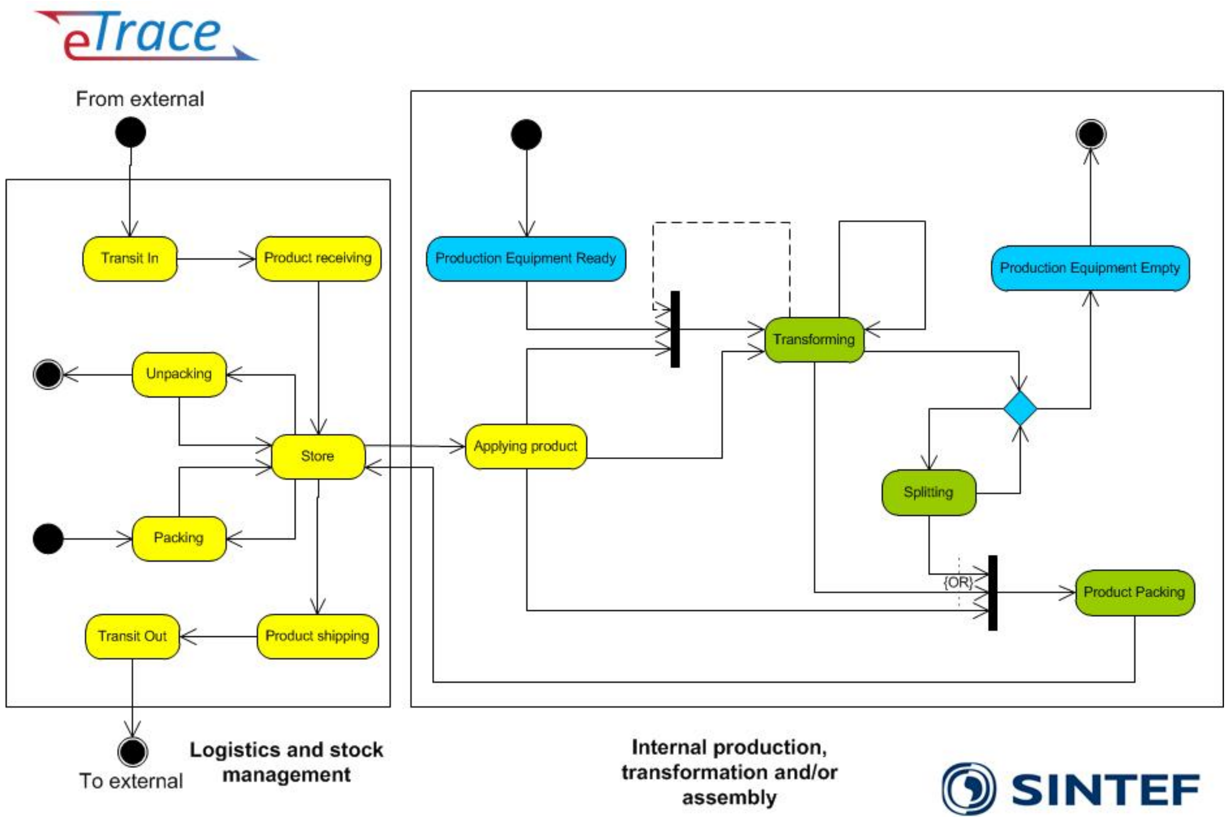
- Product name, producer (another company or another production plant within the same company)

- Tracking number in (this number can be any traceable unit like a sack, pallet, or carton)
- Production date and time, and production number
- Recommended storage temperature and temperature history (if monitored)
- Date received
- Quantity (weight, volume, number of)
- Expiration date

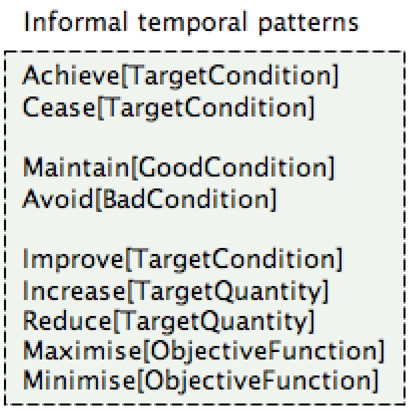
Correspondingly, products that are shipped out from the company will have data registered:

- Product name, producer
- Tracking number out (this number can be any traceable unit like a sack, pallet, or carton)
- Production date and time, and production number
- Recommended storage temperature
- Date packed
- Quantity (weight, volume, number of)
- Expiration date
- Date shipped

Recipes are used when making new, refined products. The recipes will contain which quantities to be used of every raw material (including spices), treatment methods (boiling, salting, drying, smoking, roasting, the sequence and way of mixing, etc.). Different equipment will be used in the different processes. All traceable units (pallets, boxes, etc.) that are used to move the different products between the different processing/production zones, are electronically tagged. Used equipment shall be cleaned/disinfected after given procedures and standards, before and after each production batch. All information collected through sensors, through electronic reading of RFID and all measurements of weights and item usage, are collected in an internal production system. The sensors and electronic reading points have a fixed position in the production areas. The sensors are read every minute, while the RFIDs is read when moving from one production/process-state to another. When reading the RFIDs, it is also possible to capture information about the quantities of a raw material or product involved in the capture point.



Appendix 2 – Temporal patterns for requirements



Informal temporal patterns

Achieve[TargetCondition]
Cease[TargetCondition]

Maintain[GoodCondition]
Avoid[BadCondition]

Improve[TargetCondition]
Increase[TargetQuantity]
Reduce[TargetQuantity]
Maximise[ObjectiveFunction]
Minimise[ObjectiveFunction]

Appendix 3– Code to be inspected

C.3. EncryptionServerThread.java

```
// separate thread for each client
class EncryptionServerThread extends Thread {
        // connection to the client
        private Socket socket = null;
        // constructor
        public EncryptionServerThread(Socket socket) {
                this.socket = socket;
        }

        // keep talking to the client, until the SecretEncryptor sends "Bye" to us
        public void run() {
                System.out.println("Got a new client");
                try {
                        String[] messages = new String[100];
                        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
                        BufferedReader in = new BufferedReader(
                        new InputStreamReader(socket.getInputStream()));
                        String inputLine, outputLine;
                        while ((inputLine = in.readLine()) != null) {
                                outputLine = "output>>" +
                        SecretEncryptor.encrypt(inputLine);
                                int length = outputLine.length();
                                out.println(outputLine);
                                if (outputLine.equals("Good Bye")) break;
                        }
                        out.close();
                        in.close();
                        socket.close();
                } catch (IOException e) {
                        log(e.getMessage());
                }
        }

        // Appends the given message into the log file
        private static void log(String mes) {
                try {
                        File logFile = new File("log.txt");
                        FileWriter out = new FileWriter("log.txt", false);
                        out.write(mes);
                } catch (IOException e) {
                }
        }
}
```

Appendix 4 Java Inspection Checklist

1. Variable, Attribute, and Constant Declaration Defects (VC)
1. Are descriptive variable and constant names used in accord with naming conventions?
2. Are there variables or attributes with confusingly similar names?
3. Is every variable and attribute properly initialized?
4. Could any non-local variables be made local?
5. Are all for-loop control variables declared in the loop header?
6. Are there variables or attributes that should be constants?
7. Do all attributes have appropriate access modifiers (private, protected, public)?
8. Are there static attributes that should be non-static or vice-versa?

2. Method Definition Defects (FD)
1. Are descriptive method names used in accord with naming conventions?
2. Is every method parameter value checked before being used?
3. For every method: Does it return the correct value at every method return point?
4. Do all methods have appropriate access modifiers (private, protected, public)?
5. Are there static methods that should be non-static or vice-versa?

3. Class Definition Defects (CD)
1. Does each class have appropriate constructors and destructors?
2. Do any subclasses have common members that should be in the super-class?
3. Can the class inheritance hierarchy be simplified?

4. Data Reference Defects (DR)
1. For every array reference: Is each subscript value within the defined bounds?
2. For every object or array reference: Is the value certain to be non-null?

5. Computation/Numeric Defects (CN)
1. Are there any computations with mixed data types?
2. Is overflow or underflow possible during a computation?
3. For each expression with more than one operator: Are the assumptions about order of evaluation and precedence correct?

6. Comparison/Relational Defects (CR)
1. For every boolean test: Is the correct condition checked?
2. Are the comparison operators correct?
3. Has an "&" inadvertently been interchanged with a "&&" or a "|" for a "||"?

7. Control Flow Defects (CF)
1. Will all loops terminate?
2. When there are multiple exits from a loop, is each exit necessary and handled properly?
3. Does each switch statement have a default case?
4. Are missing switch case break statements correct and marked with a comment?
5. Do named break statements send control to the right place?
6. Can any nested if statements be converted into a switch statement?
7. Are null-bodied control structures correct and marked with braces or comments?
8. Does every method terminate?

8. Module Interface Defects (MI)

1. Are the number, order, types, and values of parameters in every method call in agreement with the called method’s declaration?
2. If an object or array is passed, does it get changed, and changed correctly by the called method?

9. Comment Defects (CM)

1. Does every method, class, and file have an appropriate header comment?
2. Does every attribute, variable, and constant declaration have a comment?
3. Do the comments and code agree?
4. Are there enough comments in the code?
5. Are there too many comments in the code?

10. Input-Output Defects (IO)

1. Have all files been opened before use?
2. Are the attributes of the input object consistent with the use of the file?
3. Have all files been closed after use?
4. Are there spelling or grammatical errors in any text printed or displayed?

11. Layout and Packaging Defects (LP)

1. Is a standard indentation and layout format used consistently?
2. For each method: Is it no more than about 60 lines long?

12. Exceptions Handling (EH)

1. Are all relevant exceptions caught?
2. Is the appropriate action taken for each catch block?