UNIVERSITY *of* LIMERICK

OLLSCOIL LUIMNIGH

Faculty of Science and Engineering
Department of Computer Science
and
Information Systems

Assessment Paper

Academic Year: 2008/2009
Module Title: Leveraging Legacy Applications
Duration of Exam: 2.5 hours
Lecturer: Michael Coughlan

Semester : 2
Module Code: CS4558
% of Total Marks: 100%
Paper marked out of: 150

Instructions to Candidates.

4 questions. Attempt 3. All questions carry equal weight.

Question 1	Legacy Systems (general)	(50 marks)
Question 2	Code Renovation (practice)	(50 marks)
Question 3	Data Migration	(50 marks)
Question 4	Language Conversion (practice)	(50 marks)

Q1. Legacy Systems - General

(a) Software maintenance is normally classified by the reason(s) for which it is undertaken. Provide such a classification and briefly explain the purpose of each type of maintenance. (10 marks)

(b) With the aid of examples taken from case studies and guest lectures identify, and describe, four factors that might prompt Legacy System Modernisation. (10 marks)

(c) The modernization, or large scale modification (such as the Y2K conversion), of any large software system should be preceded by an inventory of the software portfolio. In "The Realities of Large Software Portfolios" Verhoef identifies a number of problems which such an inventory might discover. Identify, and provide a brief description for, four of these problems. (12 marks)

(d) Code Renovation is one approach to legacy system modernization. In "Revitalizing Modifiability of Legacy Assets", Veerman outlines a renovation algorithm that uses three types of code transformations: Preprocessing transformations, Mainprocessing transformations, and Postprocessing transformations. Briefly describe the purpose of each of these transformation types and mention a number of the transformation activities associated with each type. (12 marks)

(e) Most approaches to legacy system modernisation should start by thoroughly documenting the existing legacy system. In the "Commercial Software Reengineering Workbench", Sneed describes 10 basic view documents produced by the workbench. Briefly explain why so many different kinds of documentation are required. (6 marks)

Q2. Code Renovation (practice)

(a) One of the simple renovations that can be applied to legacy code is to replace delimiting full stops with explicit END delimiters (such as END-IF, END-READ etc). Briefly, and with the aid of examples, explain the problem caused by delimiting full stops and show how these problems may be solved, or highlighted, by using explicit delimiters. (8 marks)

(b) Examine the program fragment below (next page) and -

i. For each paragraph in the program state whether or not it is an *internal-procedure-like* paragraph and give the reasons why it is, or is not, such a paragraph. (12 marks)

ii. For a paragraph you have nominated as an *internal-procedure-like* paragraph, identify its *localizable* and *pseudo localizable* variables, state which is which, and state the criteria by which you so identify these variables. (14 marks)

iii. Show, by producing a skeleton of the program fragment highlighting the required changes, how the *internal-procedure-like* paragraph you selected can be converted into a parameterized Contained Subprogram. State the criteria used to identify the parameterizable variable(s) and the parameter passing mechanism(s). (16 marks)

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EnglishFootballClubs.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 PrnClubName                PIC X(20).
01 PrnCityName                PIC X(20).
01 PrnStadiumName             PIC X(20).
01 PrnCapacity                PIC ZZZ,ZZ9.

01 CityNameOut                PIC X(20).
01 StadiumNameOut             PIC X(20).
01 CapacityOut                PIC 9(6).

01 UpprClubName               PIC X(20).
01 PrnClubCount               PIC ZZ9.
01 ClubCount                  PIC 99  VALUE ZEROS.

01 ClubTable.
   02 ClubValues.
      03 FILLER PIC X(66)
         VALUE "ARSENAL             London              Emirates            060355".
      03 FILLER PIC X(66)
         VALUE "ASTON VILLA         Birmingham          Villa Park          039217".
      03 FILLER PIC X(66)
         VALUE "BLACKBURN ROVERS    Blackburn           Ewood Park          031149".
* etc
   02 FILLER REDEFINES ClubValues.
       03 Clubs OCCURS 96 TIMES
                  INDEXED BY C-Idx.
          04 ClubName     PIC X(20).
          04 CityName     PIC X(20).
          04 StadiumName  PIC X(20).
          04 Capacity     PIC 9(6).

PROCEDURE DIVISION.
GetClubNames.
    DISPLAY SPACES
    DISPLAY "Enter the name of the club :- " WITH NO ADVANCING.
    ACCEPT PrnClubName.
    MOVE PrnClubName TO UpprClubName
    IF UpprClubName = SPACES
         MOVE ClubCount TO PrnClubCount
         GO TO Wrapup.
    PERFORM GetClubDetails
    MOVE CityNameOut    TO PrnCityName
    MOVE StadiumNameOut TO PrnStadiumName
    MOVE CapacityOut    TO PrnCapacity.

DisplayClubDetails.
    IF PrnStadiumName = SPACES
       DISPLAY "Club not Found"
     ELSE
       DISPLAY "Club name = " PrnClubName
       DISPLAY "City name = " PrnCityName
       DISPLAY "Stadium   = " PrnStadiumName
       DISPLAY "Capacity  = " PrnCapacity.
       ADD 1 TO ClubCount.
    GO TO GetClubNames.

GetClubDetails.
   MOVE FUNCTION UPPER-CASE(PrnClubName) TO UpprClubName
   SET C-Idx TO 1
   SEARCH Clubs
       AT END MOVE SPACES TO StadiumNameOut, CityNameOut
              MOVE ZEROS  TO CapacityOut
       WHEN UpprClubName = ClubName(C-Idx)
              MOVE CityName(C-Idx)    TO CityNameOut
              MOVE StadiumName(C-Idx) TO StadiumNameOut
              MOVE Capacity(C-Idx)    TO CapacityOut
   END-SEARCH.

Wrapup.
   DISPLAY "No more club information required."
   DISPLAY PrnClubCount " clubs examined."
   STOP RUN.
```

Q3. Data Migration

(a) Data migration often reveals data quality problems. Briefly describe, and give examples of, the kinds of data quality problems that might be encountered in legacy system data. (12 marks)

(b) Data Consolidation involves combining existing physical structures (data stores) into a single structure. Identify, describe, and give examples of, the main schema-level and instance-level data quality problems that might have to be addressed as part of data consolidation. (14 marks)

(c) Data Warehouse Development creates an extracted relational view of the operational data and may be used as alternative to Data Migration. Briefly, and with the aid of examples, discuss the benefits and drawbacks of the Data Warehouse Development approach verses Data Migration. (12 marks)

(d) When data migration involves the core systems of an enterprise, a mechanism must be found for migrating the data that does not involve any appreciable system downtime. The Butterfly Methodology is one such mechanism. Briefly, and with the aid of diagrams, explain how the Butterfly Methodology works. (12 marks)

Q4. Language Conversion (practice)

(a) Both the "Realities of Language Conversions" and "Revitalizing Modifiability of Legacy Assets" have example COBOL code that makes use of a "Bar SECTION". Briefly describe the problem that the "Bar SECTION" is intended to solve, show how its use aids program maintenance, and identify the weakness of this solution. (12 marks)

(b) Jazillian is a company which claims to be able to convert legacy COBOL source code to Java. Their web site contains an example conversion of a small "Employee Wages" program written in COBOL. The COBOL source code and its Java conversion are given below.

In the light of the points made in the "Realities of Language Conversions" comment in some detail on how good a job the example conversion does in convincing you that Jazillian is able to accurately convert a COBOL legacy system to Java.

Your answer should address the issues of –

- Inadequacies of omission (by identifying and discussing any COBOL elements that ought to have been covered by the example conversion in order to provide convincing evidence that the Jazillian conversions are accurate and complete.) (23 marks)
- Inadequacies of commission (by identifying and discussing any inaccuracies present in the example conversion.) (15 marks)

Example COBOL source code (1 file)

Sample.Cbl

```
IDENTIFICATION DIVISION.
PROGRAM-ID SAMPLE.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.   SELECT EMPLOYEE-DATA    ASSIGN TO EMP-DAT.
                SELECT PAYROLL-LISTING  ASSIGN TO PRINTER.

DATA DIVISION.
FILE SECTION.
FD  EMPLOYEE-DATA.
01  EMPLOYEE.
    05  EMPLOYEE-NAME-IN      PICTURE X(20).
    05  HOURS-WORKED-IN       PICTURE 9(2).
    05  HOURLY-RATE-IN        PICTURE 9V99.

FD  PAYROLL-LISTING.
01  PRINT-REC.
    05                        PICTURE X(20).
    05  NAME-OUT              PICTURE X(20).
    05                        PICTURE X(10).
    05  HOURS-OUT             PICTURE 9(2).
    05                        PICTURE X(8).
    05  RATE-OUT              PICTURE 9.99.
    05                        PICTURE X(6).
    05  WEEKLY-WAGES-OUT      PICTURE 999.99.

WORKING-STORAGE SECTION.
01  ARE-THERE-MORE-RECORDS    PICTURE XXX VALUE 'YES'.

PROCEDURE DIVISION.
100-MAIN-MODULE.
    OPEN INPUT EMPLOYEE-DATA
         OUTPUT PAYROLL-LISTING
    PERFORM UNTIL ARE-THERE-MORE-RECORDS = 'NO '
        READ EMPLYEE-DATA
           AT END
                 MOVE 'NO ' TO ARE-THERE-MORE-RECORDS
           NOT AT END
               PERFORM 200-WAGE-ROUTING
        END-READ
    END-PERFORM
    CLOSE EMPLOYEE-DATA
    CLOSE PAYROLL-LISTING
    STOP RUN.

200-WAGE-ROUTINE.
    MOVE SPACES TO PRINT-REC
    MOVE EMPLYEE-NAME-IN TO NAME-OUT
    MOVE HOURS-WORKED-IN TO HOURS-OUT
    MOVE HOURLY-RATE-IN TO RATE-OUT
    MULTIPLY HOURS-WORKED-IN BY HOURLY-RATE-IN
            GIVING WEEKLY-WAGES-OUT
    WRITE PRINT-REC.
```

Example translated Java Code (3 files – Sample.Java, Employee.Java, PrintRec.Java)

Sample.Java

```
/*
IDENTIFICATION DIVISION ;
 PROGRAM_ID ;
 SAMPLE ;
*/
import jazillian.runtime.cobol.*;
import java.io.IOException;
public class Sample {
    public String areThereMoreRecords = "YES";
    public Employee employee = new Employee();
    public PrintRec printRecord = new PrintRec();
    CobolInputStream employeeData;
    CobolOutputStream payrollListing;
    public static void main(String[] args) {
        Sample instance = new Sample();
        instance.initialParagraph();
        instance.mainModule();
    }
    void initialParagraph() {
        payrollListing = CobolOutputStream.newInstance("PRINTER");
        employeeData = CobolInputStream.newInstance("EMP_DAT");
    }
    void mainModule() {
        while (!(areThereMoreRecords.equals("NO "))) {
            employee = AppUtils.readEMPLOYEE(employeeData);
            if (employeeData.isAtEnd()) {
                areThereMoreRecords = "NO ";
            }
            else {
                wageRoutine();
            }
        }
        try {
            employeeData.close();
            payrollListing.close();
        }
        catch (IOException closeException) {
            System.err.println("IO Exception on close call:" + closeException);
        }
    }
    void wageRoutine() {
        printRecord = new PrintRec();
        printRecord.nameOut = employee.employeeNameIn;
        printRecord.hoursOut = employee.hoursWorkedIn;
        printRecord.rateOut = employee.hourlyRateIn;
        printRecord.weeklyWagesOut = employee.hourlyRateIn *
     employee.hoursWorkedIn;
        AppUtils.writePRINT_REC(payrollListing, printRecord);
    }
}
```

Employee.Java

```
class Employee {
    String employeeNameIn;
    int hoursWorkedIn;
    float hourlyRateIn;
}
```

PrintRec.Java

```
class PrintRec {
    String nameOut;
    int hoursOut;
    float rateOut;
    float weeklyWagesOut;
}
```