Student ID: .....

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2013

TRIMESTER 1

COMP 102
INTRODUCTION TO COMPUTER PROGRAM DESIGN
(improved from original)

Time Allowed: THREE HOURS ******** WITH SOLUTIONS **********

Instructions: Attempt ALL Questions.

The exam will be marked out of 180 marks.

Silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted.

Printed foreign language dictionaries are permitted.

Java Documentation will be provided with the exam script

No other material is permitted.

Answer in the appropriate boxes if possible — if you write your answer elsewhere, make it clear where your answer can be found.

There are spare pages for your working and your answers in this exam, but you may ask for additional paper if you need it.

Questions

	Marks
1. Understanding Java	[59]
2. Defining a Class	[12]
3. Files	[17]
4. Event driven input	[13]
5. ArrayLists of Objects	[30]
6. Interface classes	[14]
7. 2D Arrays	[15]
8. Debugging loops	[20]

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 1. Understanding Java [59 marks]

(a) [4 marks] What will the following fragment of Java print out?

```
int x = 1;
int y = 20;
while (y > 0){
    UI.println (x + "  :  " + y);
    x = x * 2;
    y = y-5;
}
UI.println (x + " done");
```

x: [ ] y: [ ]

```
1  :  20
2  :  15
4  :  10
8  :  5
16 done
```

(b) [6 marks] Many lines of the following broken method have a syntax error that the compiler would complain about. Circle, and briefly describe four errors

```
public String void broken(double a, double b){ // must not have two return types ;
    if (a ==> b){                               // ==> should be >= or ==
        a++ = b + 1;                            // can't assign to a++, must be a variable
        break;                                  // break must be inside a loop (while or for)
    else {                                      // need a closing }
        while (b = 1){                          // condition must be a boolean, eg b == 1
            b = b + 1;
        }
    }
    return "yes";
}
```

(Question 1 continued on next page)

(Question 1 continued)

(c) [6 marks] The expressionTest method below has one parameter and performs three tests on the parameter, printing out which tests passed.
Note that there are no **else**'s.

```
public void expressionTest(int value){

    if ( value > 0 && value <= 5 ) {
        UI.print("first ");
    }

    if ( value < 7 || value > 10 ) {
        UI.print("second ");
    }

    if ( ! (value == 0) ) {
        UI.print("third ");
    }

    UI.println();
}
```

What would the following calls to expressionTest print out?

expressionTest(5); ⟹ first second third

expressionTest(7); ⟹ third

expressionTest(0); ⟹ second

(Question 1 continued on next page)

(Question 1 continued)

(d) [5 marks] Complete the following totalCharge method to compute and return the total charge for an online order. It should have two parameters (doubles) which are the cost of the item and the cost of shipping. The total charge (including GST) is 1.15 times the sum of the item cost and the shipping cost.

```
public double totalCharge (double items, double shipping){

    return (items + shipping) * 1.15;


}
```

(e) [7 marks] What will be printed in the text pane if the following table method is called with the argument 5:

```
public void table (int size){
    for (int x = 1; x < size; x++){
        for (int y = x+1; y < size; y++){
            UI.print(x + " vs " + y + ", ");
        }
        UI.println("stop");
    }
    UI.println("done");
}
```

x: ☐ y: ☐

```
1 vs 2, 1 vs 3, 1 vs 4, stop
2 vs 3, 2 vs 4, stop
3 vs 4, stop
stop
done
```

(Question 1 continued on next page)

(f) [7 marks] Suppose the file `scores.txt` contains the text:

```
Otago 5 soccer
Auckland 8 hockey
Wellington 7 cycling
```

What will the following processScores method print out?

```
public void processScores(){
    try{
        Scanner scan = new Scanner (new File("scores.txt"));
        int number = 1;
        while ( scan.hasNext() ){
            String first = scan.next();
            UI. println ( first );
            number = number + scan.nextInt();
            if (scan.hasNextInt()){
                UI. println ("score: " + scan.nextInt());
            }
            else {
                UI. println ("game: " + scan.next());
            }
        }
        scan.close();
        UI. println ("Found: " + number);
    }
    catch(IOException e){UI.println("File reading failed");}
}
```

number:

```
Otago
game:  soccer
Auckland
game:  hockey
Wellington
game:  cycling
Found:  21
```

(Question 1 continued on next page)

(Question 1 continued)

(g) [10 marks] Consider the following modifyArray method which compares numbers in an array and changes the values.

```
public void modifyArray(int [ ] data){
    for( int i=1; i<data.length; i++){
        if ( data[i−1] < data[i] ){
            data[i] = data[i−1];
            data[i−1] = −99;
        }
    }
}
```

i: ☐

Suppose that the variable sizes is defined as follows:

```
int [ ] sizes = new int [ ]{4, 3, 5, 2, 6, 4, 1};
```

sizes:

4	3	5	2	6	4	1
0	1	2	3	4	5	6

Show the contents of sizes after calling modifyArray(sizes);

sizes:

4	-99	3	-99	-99	2	1
0	1	2	3	4	5	6

(Question 1 continued on next page)

(Question 1 continued)

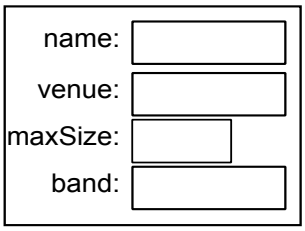
(h) [14 marks] The PartyPlan class below defines PartyPlan objects, which store information about planned parties. Each object has four fields to store the name of the party, the venue the party will be held at, the maximum size of the party, and the band. The class defines a constructor and three methods.

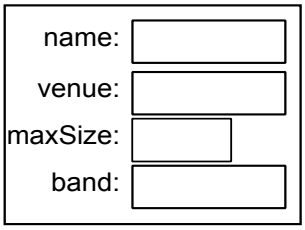
```
public class PartyPlan{
    private String name;
    private String venue;
    private int maxSize;
    private String band;

    public PartyPlan(String name, String venue, int size){
        this.name = name;
        this.venue = venue;
        this.maxSize = size;
    }

    public void print(){
        UI.printf("%s at %s (max %d)", this.name, this.venue, this.maxSize);
        if (this.band != null){ UI.println(" with "+ this.band); }
        else { UI.println(); }
    }

    public void setBand (String newBand){
        if (this.band != null ){
            UI.println("Bumped "+ this.band);
        }
        this.band = newBand;
        UI.println("Set band to "+ newBand);
    }

    public void mergeIntoParty(PartyPlan other){
        if (this.maxSize > other.maxSize){
            UI.println("No room to merge into " + other.name);
        }
        else {
            other.setBand(this.band);
            this.maxSize = 0;
            this.band = null;
            UI.println("Merged");
        }
    }
}
```

(Question 1 continued)

What will the following testPartyPlan method print out?

```
public static void testPartyPlan(){

    PartyPlan partyG = new PartyPlan("Glens Graduation", "Gleam Park", 30);
    partyG.print ();

    PartyPlan partyB = new PartyPlan("Bills Birthday", "Basin Rd", 3);
    partyB.setBand("Banyan");
    partyB.setBand("Beanfield");
    partyB.print ();


    partyG.setBand("Ginos");
    partyB.mergeIntoParty(partyG);

    partyG.print ();

    partyG.mergeIntoParty(partyB);
    partyB.print ();

}
```






```
Glens Graduation at Gleam Park (max 30)
Set band to Banyan
Bumped Banyan
Set band to Beanfield
Bills Birthday at Basin Rd (max 3) with Beanfield
Set band to Ginos
Bumped Ginos
Set band to Beanfield
Merged
Glens Graduation at Gleam Park (max 30) with Beanfield

No room to merge into Bills Birthday
Bills Birthday at Basin Rd (max 0)
```

Question 2. Defining a Class [12 marks]

Complete the TrainingCourse class on the facing page which stores information about courses offered by a Technical Training Academy.

A TrainingCourse object should contain four fields:

- name, which contains the name of the course (*e.g.* "Cabinet Making")
- weeks, which contains the length of the course in weeks (*e.g.* 8)
- cost, which contains the cost of the course per student (*e.g.* 480.50)
- enrolment, which contains the number of students enrolled in the course, initially 1 when the course is first created.

TrainingCourse should have a constructor that takes one String parameter and one integer parameter and stores them in the name and weeks fields.

TrainingCourse should have four methods:

- setCost, which is passed the cost of the course, and sets the cost field to contain this number.
- addStudents, which is called when students enrol in the course, and is passed the number of students who are enrolling. It should add the argument to the enrolment field.
- totalRevenue, which returns the total revenue of the course (the cost times the number of students enrolled).
- print(), which prints out the name of the course, the length in weeks, and the number of students enrolled, in the format

```
"Cabinet Making (8 weeks) 23 students"
```

The header of the constructor and one of the methods are given.

(Question 2 continued on next page)

(Question 2 continued)

```
public class TrainingCourse{
    // fields

    private String name;
    private int weeks;
    private double cost;
    private int enrolment = 1;
    // constructor
    public TrainingCourse(String nm, int wks){


        this.name = nm;
        this.weeks = wks;
    }

    // methods
    public void setCost(double c){

        this.cost = c;
    }


    public void addStudents(int num){
        this.enrolment = this.enrolment + num;
    }

    public double totalRevenue(){
        return this.cost * this.enrolment;
    }

    public void print(){
        UI.println(this.name + "  (" + this.weeks + " weeks)  " + this.enrolment + " student
    }









}
```

Question 3. Files [17 marks]

Suppose the file `allcars.txt` contains data about cars at a car dealer. Each line of the file contains the model, the year, and the price of a car. For example, the following might be the first six lines of the file.

```
Starlet     2001   8175
Demio       2006  14295
Axela       2008  17875
Swift       2008  11875
Accord      2004  13875
Mondeo      2009  15995
```

Assume the model name is always a single word.

(a) [7 marks] Complete the following printRecent method, whose parameter is a year. printRecent should read the file and print out the model and price of each car from the specified year or later.

For example, on the data above, printRecent(2006); should print

```
Cars from 2006 onwards:
 Demio $14295
 Axela $17875
 Swift $11875
 Mondeo $15995
```

```
public void printRecent(int minYear){
    UI.println("Cars from "+ minYear + " onwards:");
    try{
        Scanner sc = new Scanner(new File("allcars.txt:"));

        while (sc.hasNext()){
            String model = sc.next();
            int year = sc.nextInt();
            double price = sc.nextDouble();
            if (year >= minYear){
                UI.printf("%s $%.0f%n", model, price);
            }
        }

        sc.close();
    }catch(IOException e){UI.println("fail");}
}
```

(Question 3 continued)

(b) [10 marks] The file is supposed to have all the cars listed in increasing order of year, but sometimes it is edited wrongly. Complete the following checkOrdered method which should check whether the cars are in order by year. It should print "All in order" if there are no cars out of order. Otherwise, it should print out each car whose year is earlier than the year of the immediately preceding car. You may assume there is at least one car in the list.

For example, if the file contained the list on the facing page, checkOrdered() should print out

```
Checking for cars out of order:
Accord 2004 13875
```

```
public void checkOrdered (){
    UI. println ("Checking for cars out of order:");
    try{
        Scanner sc = new Scanner (new File ("allcars.txt"));

         boolean OK = true;
         sc.next ();
         int  prevYr = sc. nextInt ();
         sc.next ();
         while (sc.hasNext()){
              String  model = sc.next();
              int  year = sc. nextInt ();
              double price = sc.nextDouble();
              if  (year < lastYear){
                   UI. println (  model + "   " + year + "   " + price);
                   OK = false;
              }
              lastYear  = year;
         }
         if  (OK){
              UI. println ("All in order");
         }




        sc.close ();
    }
    catch(IOException e){UI.println("Fail:  " + e);}
}
```

Question 4. Event Driven Input [13 marks]

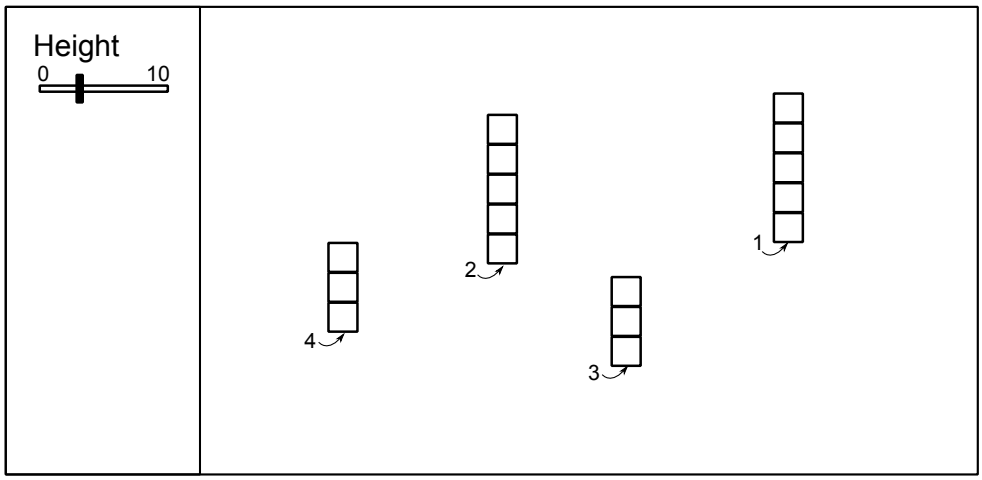
Complete the BrickLayer program on the facing page so that it allows the user to place columns of blocks on the graphics pane. The program should have a slider called "Height". It should also listen to the mouse.

Every time the user moves the slider, the program should remember the new value.

When the user releases the mouse in the graphics pane, the program should draw a column of square blocks at the release point. The center of the base of the column should be at the point, and the number of blocks in the column should be the last height that the user set with the slider (or 2 if the user hasn't set the slider yet).

The diagram below shows what the program should do if the user

- changed the slider to 5,
- clicked the mouse at the positions labeled 1 and 2
- changed the slider to 3,
- clicked the mouse at the positions labeled 3 and 4



Hint: The sliderPerformed method works very like the textFieldPerformed method: the value parameter is passed the number that the slider was set to. See the documentation.

(Question 4 continued)

```
public class BrickLayer implements UISliderListener, UIMouseListener{
    // fields
    private static final double size = 20;   // width and height of the blocks.
    private int columnHeight = 2;


    public BrickLayer(){
        UI.setMouseListener(this);
        UI.addSlider("Height", 0, 10, this);


    }
    public void sliderPerformed(String slider, double value){
        if (slider.equals("Height")){
            this.columnHeight = (int) value;
        }


    }
    public void mousePerformed(String action, double x, double y) {
        if (action.equals("released")){
            double left = x-this.size/2;
            double top = y-this.size;
            for (int i=0; i<this.columnHeight; i++){
                UI.drawRect(left, top, this.size, this.size);
                top = top-this.size;
            }
        }









    }
```

Question 5. ArrayLists of Objects [30 marks]

This question concerns the PastStudentDatabase program to keep track of the past students of a university. The PastStudentDatabase class (first few lines are shown on the facing page) stores details of each past student in an ArrayList of Student objects.

The Student class, shown below, defines Student objects that store information about individual past students, including their year of graduation and the majors they graduated with.

```
public class Student{
    // fields
    private String name;
    private int ID;
    private int gradYear;    // year graduated
    private ArrayList<String> majors; // list of majors

    // constructor
    public Student(String name, int ID, int gradYear){
        this.name = name;
        this.ID = ID;
        this.gradYear = gradYear;
        this.majors = new ArrayList<String>();
    }
    //methods
    public int getID() {
        return this.ID;
    }
    public int getGradYear(){
        return this.gradYear;
    }
    public boolean hasMajor(String major) {
        return this.majors.contains(major);
    }
    public void addMajor(String major) {
        this.majors.add(major);
    }
    public void printDetails (){
       UI.print (this.name + "("+ this.ID +") year: " + this.gradYear );
       UI.print (" Majors: ");
       for(String major : this.majors){ UI.print (major+" "); }
       UI.println ();
    }
}
```

(Question 5 continued)

First few lines of the PastStudentDatabase class:

```
public class PastStudentDatabase{

    private ArrayList <Student> students = new ArrayList<Student>();

    public void listPastStudents(){
        for (Student st : this.students){
            st.printDetails();
        }
    }
       :
```

(a) [6 marks] Complete the following findEarliest method of the PastStudentDatabase class. It should find the Student who graduated earliest and print out their details. If there are several students who are equally early, then the method should print just the first one it finds. Assume the list contains at least one student.

```
public void findEarliest(){
    Student earliest = this.students.get(0);
    for (Student student : this.students) {
        if (student.getGradYear() < earliest.getGradYear()){
            earliest = student;
        }
    }
    earliest.printDetails();

// Alternative (not as neat):

    int minYear = Integer.MAX_VALUE;
    int minStIndex = 0;
    for (int i=0; i<this.students.size(); i++) {
        if (this.students.get(i).getGradYear() < minYear){
            minYear = this.students.get(i).getGradYear();
            minStIndex = i;
        }
    }
    this.students.get(minStIndex).printDetails();

}
```

(Question 5 continued on next page)

(Question 5 continued)

(b) [8 marks] Complete the following addStudent method with one parameter – a Student. The method should check whether the list already has a student with the same ID as the parameter. If so, the method should print the message `"Student already in database"` and not modify the list. If the student is not present, the method should add the student and print the message `"Student added"`.

```
public void addStudent(Student student){
    int ID = student.getID();
    for (Student st : this.students){
        if (st.getID() == ID){
            UI.println("Student already in database");
            return;
        }
    }
    this.students.add(student);
    UI.println("Student added");




}
```

(c) [8 marks] Complete the following constructMajorList method which has one parameter — the name of a major. It should construct and return an ArrayList of all the students in the database who graduated with the specified major.

```
public ArrayList<Student> constructMajorList(String major){
    ArrayList<Student> answer = new ArrayList<Student>();
    for (Student student : this.students){
        if (student.hasMajor(major)){
            answer.add(student);
        }
    }
    return answer;
// Alternative loop:
    for (int i=0; i<this.students.size(); i++){
        if (this.students.get(i).hasMajor(major)){
            answer.add(this.students.get(i));
        }
    }

}
```

(Question 5 continued on next page)

(Question 5 continued)

(d) [8 marks] Complete the following loadDB method in the PastStudentDatabase class. Its parameter is a *String* specifying the name of a file containing data about past students. The method should clear the students field, then read the file, constructing a Student object for each student and adding the Student object to the list of students.

The format of the file is that each student is described on two lines. The first line contains the student's ID and name. The second line contains the graduation year and the list of majors. For example, the following six lines describe three students:

```
500432 Jason Smith
2004 COMP MATH
500231 Bryce John Sillars
1999 BIOL
500321 Jane Hollistan
2007 LING COMP INFO
```

```
public void loadDB(String filename){
    try{
        this.students.clear ();
        Scanner sc = new Scanner (new File (filename));
        while(sc.hasNext()){
            int ID = sc.nextInt ();
            String name = sc.nextLine().trim (); //trim removes space at beginning
            int year = sc.nextInt ();

            Student st = new Student(name, ID, year);
            this.students.add(st);

            //add majors: check not at end of file & not next student
            while (sc.hasNext() && !sc.hasNextInt()){
                st.addMajor(sc.next());
            }
        }

 // alternative way of reading name (space between words, no spaces at ends)
            String name = sc.next();
            while (!sc.hasNextInt()) { name = name + "  " + sc.next(); }

 // alternative way of reading majors
            Scanner line = new Scanner(sc.nextLine());
            while (!line.hasNext()) { st.addMajor(line.next ()); }

    }
    catch(IOException e){UI.println("Fail: " + e);}
}
```

Question 6. Interface classes [14 marks]

The FungiForest game involves a player exploring a forest made up of different kinds of magical fungus: mushrooms, toadstools, puffballs, and lichen. The different types of fungus behave in different ways. They can all be drawn, and they can all grow (though toadstools only grow upwards, and lichen only grows sideways). They become activated in different situations, have different strengths, and can respond to attacks in different ways.

However, every fungus will have the following things:

- fields to store values such as the size, the current strength, and whether the fungus is activated.
- a constructor for placing a new fungus at a given position in the forest
- a draw method that draws the fungus.
- an isActivated method that returns a boolean value representing whether the item is activated or not.
- a getStrength method that returns a number specifying the current strength of the fungus
- an attack method with one parameter (a String specifying the weapon) that may damage the fungus.
- a grow method, with one parameter, that causes the fungus to grow the specified amount.

The FungiForest program uses an interface class called Fungus to define the type of all the different kinds of fungus.

(a) [7 marks] Complete the following definition of the Fungus interface class.

```
public interface Fungus {
    public void draw();

    public boolean isActivated();

    public double getStrength();

    public void attack(String weapon);

    public void grow(double amt);

  // Note: must not have any fields
  // Note: must not have a constructor
  // Note: must not have any method bodies


}
```

(Question 6 continued on next page)

(Question 6 continued)

(b) [7 marks] The following Toadstool class describes Toadstool objects, and provides several methods for Toadstools. In the game, toadstools are very simple items that grow vertically, are always activated, and lose half their current strength when attacked (with any weapon).

However, Toadstool is not correctly written to make every Toadstool object also be a Fungus. Modify the definition of Toadstool below so that a Toadstool is also a Fungus.

```
public class Toadstool {X   implements Fungus {
    private double strength = 10.0;
    private double posX;
    private double posY;
    private Color col = new HSBColor(Math.random(), 1.0f, 1.0f);
    private double width = 30.0;
    private double height = 10.0;

    public Toadstool(double x, double y){
        this.posX = x;
        this.posY = y;
    }
    public void grow(double amt){ this.height += amt;}

    public void draw(){
        UI.setColor(this.col);
        UI.drawOval(posX-this.width/2, posY-this.height/2, this.width, this.height);
    }


    // Note: all the methods in Fungus must be defined.

    public boolean isActivated(){
        return true;
    }

    public double getStrength(){
        return this.strength;
    }

    public void attack(String weapon){
        this.strength = this.strength / 2;
    }



}
```

Question 7. 2D Arrays [15 marks]

A matrix is a 2D rectangular array of numbers. There are many applications of matrices and many different operations on them. For this question, you are to implement three methods that operate on matrices.

(a) [8 marks] Complete the following printMatrix method that prints out the values in a matrix, lined up in rows and columns. For example, if the matrix has two rows and 3 columns, it might print it as follows:

```
  6.23   15.41  -13.45
-27.18   -3.87   85.00
```

The method should print each value with two decimal places, ensuring that there are at least two spaces between numbers. You may assume that every number in the matrix can be printed with at most 6 characters.

Assume that the first index represents the row, and the second index represents the column.

```
public void printMatrix(double [ ][ ] matrix){
    for (int r =0; r<matrix.length; r++){
        for (int c =0; c<matrix[r].length; c++){
            UI.printf("  %6.2f", matrix[r][c]);
        }
        UI.println();
    }
//  alternative
    for (double [ ] row : matrix){
        for (double val : row){
            UI.printf("%8.2f", val);
        }
        UI.println();
    }

}
```

(Question 7 continued on next page)

(Question 7 continued)

(b) [7 marks] The transpose of a matrix is a matrix in which the rows have been swapped with the columns. For example, the transpose of

$$\begin{pmatrix} 2 & 11 & -16 \\ 8 & 5 & 16 \\ 12 & -7 & 62 \\ 1 & 30 & 8 \end{pmatrix}$$

is

$$\begin{pmatrix} 2 & 8 & 12 & 1 \\ 11 & 5 & -7 & 30 \\ -16 & 16 & 62 & 8 \end{pmatrix}$$

If the matrix is not square (*i.e.*, has different number of rows and columns), then the transpose will have different dimensions from the origial matrix.

Complete the following transpose method which is passed one matrix, and will construct and return a new matrix that is the transpose of its argument.

```
public double[][] transpose(double[][] matrix){
    int rows = matrix.length;
    int cols = matrix[0].length;
    double[][] ans = new double[cols][rows];
    for (int r=0; r<rows; r++){
        for (int c=0; c<cols; c++){
            ans[c][r] = matrix[r][c];
        }
    }
    return ans;









}
```

Question 8. Debugging Loops [20 marks]

The following removeDuplicates method is intended to remove any duplicate values from an array of strings, by replacing all but the first of any duplicate values by null.

For example, given the array:

pets:	"fox"	"kea"	"fox"	"cat"	"tui"	"fox"	"tui"	"cat"
	0	1	2	3	4	5	6	7

removeDuplicates(pets) should change the array to be

pets:	"fox"	"kea"	null	"cat"	"tui"	null	null	null
	0	1	2	3	4	5	6	7

The following version of removeDuplicates has multiple errors.

```
/** Replace duplicate values by null, leaving only first occurrence of each distinct value
    (Broken) */
public void removeDuplicates (String[ ] array){
    for (String val : array){
        for (int i=0; i<array.length; i++){
            if (val.equals(array[i])){
                array[i] = null;
            }
        }
    }
}
```

(Question 8 continued on next page)

(Question 8 continued)

(a) [7 marks] If this version of removeDuplicates (with the errors) were called with the pets array on the facing page:

```
removeDuplicates(pets);
```

it would crash and report an exception. What error would it report and what would the contents of the pets array be at that point ?
Hint: show your working.

Exception: null pointer exception (trying to call equals on null).

Note that val.equals(array[i]) is fine when array[i] is null, because you can safely pass null, as an argument to equals, but it crashes when val is null, because you can't call any method on null

pets:

null	null	null	"cat"	"tui"	null	"tui"	"cat"
0	1	2	3	4	5	6	7

(b) [3 marks] Identify and briefly describe one of the errors in this version of removeDuplicates.

The inner for loop will remove ALL occurrences of each value, including the first, even values with no duplicates.

OR

If there are any duplicates, the outer loop will get to a null when it gets to the 2nd instance of the duplicated value, and will try to call equal on the null, which will cause an exception

(Question 8 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 8 continued)

(c) [10 marks] Write a correct version of removeDuplicates.

```
/** Replace duplicate values by null, leaving only first occurrence of each distinct value */
public void removeDuplicates(String[] array){

        for (int j=0; j<array.length; j++){
            String val = array[j];
            if (val != null){
                for (int i=j+1; i<array.length; i++){
                    if (val.equals(array[i])){
                        array[i] = null;
                    }
                }
            }
        }
// Alternative (less efficient because it does unnecessary checking):
        for (String val : array){
            if (val != null){
                boolean first = true;
                for (int i=0; i<array.length; i++){
                    if (val.equals(array[i])){
                        if (first) { first = false; }
                        else { array[i] = null; }
                    }
                }
            }
        }


}
```

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

****