



**EXAMINATIONS — 2010**

**MID-YEAR**

**COMP 102  
INTRODUCTION TO  
COMPUTER PROGRAM  
DESIGN**

**Time Allowed:** 3 Hours

**Instructions:** Attempt ALL Questions.

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

The exam will be marked out of 180 marks.

Non-programmable calculators without a full alphabetic key pad are permitted.

Non-electronic foreign language dictionaries are permitted.

Java Documentation will be provided with the exam script

There are spare pages for your working and your answers in this exam.

**Questions**

	<b>Marks</b>
1. Understanding Java	[63]
2. Files	[27]
3. Arrays of Objects	[22]
4. Event driven input	[20]
5. 2D Arrays	[26]
6. Interface classes	[10]
7. Debugging loops	[12]

Note: this version has several errors and ambiguities corrected — it is not identical to the exam that was actually sat.

**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**

[63 marks]

(a) [3 marks] What will the following fragment of Java print out?

```

int x = 2;
int y = 4;
x = x + 3;
y = y * x;
Ul.println ("x=" + x + " y=" + y);
x = y;
y = x;
Ul.println ("x=" + x + " y=" + y);

```

(b) [6 marks] Consider the following choice method (note the **if**'s and **else**'s carefully):

```

public void choice(int x, int y) {
    if (x < 15 || y > 20){
        Ul.print ("1st ");
    }
    if (x >= 8 && y > 10) {
        Ul.print ("2nd ");
    }
    else if (x < 10 && y==8){
        Ul.print ("3rd ");
    }
    else {
        Ul.print ("4th ");
    }
    Ul.println ();
}

```

What would the following calls to choice print out?

```

choice(8, 8) ==>
choice(20, 20) ==>
choice(8, 20) ==>

```

(Question 1 continued on next page)

**(Question 1 continued)**

**(c)** [5 marks] What will the following fragment of Java print out?

```
int j = 4;
while (j > 1){
    int k=1;
    while (k <= j){
        UI.printf (" (%d, %d) ", j, k);
        k++;
    }
    j = j/2;
}
UI.println ("Done");
```

**(d)** [5 marks] What will the following fragment of Java print out if the user enters the following in response to the prompt:

after seven 15 the 5 flies, the worm, and the bug

```
UI.println ("Input: ");
while ( true ){
    String word = UI.next();
    if ( UI.hasNextInt() ) {
        UI.println (word + " " + UI.nextInt ());
    }
    else if (word.equals("the") ) {
        break;
    }
    else {
        UI.println ("word: " + word);
    }
}
UI.println ("Done");
```

Input: after seven 15 the 5 flies, the worm, and the bug

**(Question 1 continued)**

(e) [7 marks] Suppose the variable `words` is declared and initialised as follows:

```
String [ ] words = new String [ ] { "dog", "bee", "fox", "cat", "ant", "eel" };
```

words:	dog	bee	fox	cat	ant	eel
	0	1	2	3	4	5

What will the following code fragment print out?

```
for( int j = 0; j < words.length-1; j++){
    Ul.print(words[j+1]+ " ");
}
Ul.println ();
for ( int k = 1; k <= words.length / 2; k++){
    Ul.printf ("%s<->%s ", words[k], words[words.length-k]);
}
Ul.println ();
```

(f) [7 marks] Suppose that the variable `words` is declared and initialised as before:

```
String [ ] words = new String [ ] { "dog", "bee", "fox", "cat", "ant", "eel" };
```

The following extract method has a parameter that is an array of `String` and it returns an array of `int`. Show the array it will return if it is called on `words`: `extract(words)`;

```
public int[ ] extract( String [ ] wds){
    int [ ] ans = new int [wds.length];
    for( int k = 1; k < wds.length; k++){
        wds[k] = wds[k] + wds[k-1];
        ans[k] = wds[k].length ();
    }
    return ans;
}
```

extract(words) ⇒						
	0	1	2	3	4	5

(Question 1 continued on next page)

**(Question 1 continued)**

The Shipment class on the facing page defines Shipment objects, which have three fields to store their destination, current load, and a load limit. The class defines three methods on Shipments and a test method.

**(g)** [6 marks] If the test method is called, what will it print out?

A :
B :
C :
D :
E :
F :
G :

**(h)** [4 marks] Write an additional method for the Shipment class called space with no parameters which returns the number of units that could be added to the Shipment without the load rising to the limit.

--

(Question 1 continued on next page)

**(Question 1 continued)**

```

public class Shipment{
    private String destination;
    private int load;
    private final int limit ;

    public Shipment(String dest, int lim){
        this.destination = dest;
        this.load = 0;
        this.limit = lim;
    }
    public String toString(){
        return (this.load + "/" + this.limit + " to " + this.destination);
    }
    public void addToShipment (int units){
        if (this.load + units < this.limit ) {
            this.load = this.load + units;
        }
    }
    public Shipment split(){
        if ( this.load > this.limit /2){
            int half = this.load/2;
            this.load = this.load - half;
            Shipment newShip = new Shipment(this.destination, this.limit);
            newShip.addToShipment(half);
            return newShip;
        }
        return null;
    }
    public static void test(){
        Shipment s1 = new Shipment("AKL", 45);
        Ul.println ("A: " + s1.toString());

        s1.addToShipment(100);
        Ul.println ("B: " + s1.toString());
        s1.addToShipment(25);
        Ul.println ("C: " + s1.toString());
        s1.addToShipment(20);
        Ul.println ("D: " + s1.toString());

        Shipment s2 = s1.split ();
        Ul.println ("E: " + s1.toString());
        Ul.println ("F: " + s2.toString());
        s2.addToShipment(10);
        Ul.println ("G: " + s2.toString());
    }
}

```

**(Question 1 continued)**

(i) [6 marks] Suppose the file `flights.txt` contains the following text:

```
AKL 160 145
WLG MAS 32 30
AKL 200 199
CHC DNN 100 78
```

What will the following `printFile` method print out?

```
public void printFile (){
    try{
        Scanner scan = new Scanner (new File("flights.txt"));
        int tot = 0;
        while ( scan.hasNext() ){
            String str = scan.next();
            if (scan.hasNextInt()){
                int incr = scan.nextInt() - scan.nextInt();
                UI.println (incr);
                tot = tot + incr;
            }
            else {
                UI.println (str);
            }
        }
        UI.println ("Tot = " + tot);
        scan.close();
    }
    catch(IOException e){UI.println("File reading failed");}
}
```

(Question 1 continued on next page)

**(Question 1 continued)**

(j) [6 marks] Complete the following `sum` method. `sum` has one parameter – an array of `int` – and should return the sum of the values in the array.

```
public double sum(int [ ] data){
```

```
}
```

(k) [8 marks] Complete the following definition of a `Bus` class. `Bus` objects should have two fields called `company` and `capacity`. `company` should hold a `String` specifying the company that owns it, and `capacity` should hold a number specifying the number of seats on the bus.

The class should have a constructor that takes two arguments and sets the fields to its arguments. The class should have one method called `toString` (with no parameters) that returns a `String` describing the object (which should contain the values of both fields).

```
public
```

```
}
```

## Question 2. Files

[27 marks]

Suppose the file `CarData.txt` contains data about secondhand cars. Each line of the file contains information about one car: the make, the model, the year, and the price. For example, the following might be the first eight lines of the file.

```
Honda Civic 1994 29699
Ford Fiesta 1993 24799
Honda Jazz 2006 27599
Nissan Maxima 1997 6999
Toyota Prius 1994 23099
Ford Focus 2008 14399
Ford Mondeo 1993 20299
Toyota Prius 1994 15499
```

(a) [7 marks] Complete the following `listMake` method, whose parameter is the name of a make (e.g. `Honda`). `listMake` should print out the details (model, year, price) of each car in the file of the given make. For example, on the data above, `listMake("Honda")`; should print

Cars by Honda

```
Civic 1994 29699
Jazz 2006 27599
```

```
public void listMake(String make){
    String filename = "CarData.txt";
    UI.println ("Cars by " + make);
    try{

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

(Question 2 continued on next page)

**(Question 2 continued)**

**(b)** [10 marks] Complete the following `yearOfMostCars` method which should print out the year for which there are the most cars listed in the file, along with the number of cars for that year. For example, if the file only had the eight lines in the example above, `yearOfMostCars()` should print out

```
Max year = 1994: 3 cars
```

since there are more 1994 cars than any other year. You may assume there is just one year that has the maximum number of cars.

```
public void yearOfMostCars(){
    String filename = "CarData.txt";
    try{

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

(Question 2 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 2 continued)**

(c) [10 marks] Suppose that some of the car model names consisted of more than one word. For example, the file might contain entries such as

```
Honda      Jazz      1994  29699
Ford       Crown Victoria  1985  15799
Nissan     Maxima   1997   6999
Chevrolet  Monte Carlo    1979  43099
Jaguar     Series III E Type Convertible  1983  54999
```

Complete the following `oldestCar` method which will print out the year and price of the oldest car in the file, assuming that the car model names may consist of any number of words. You may assume that there will be at least one car in the inventory file, and that none of the model names contain numbers.

```
public void oldestCar(){
    String filename = "CarData.txt";
    try{

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

### Question 3. Arrays of Objects

[22 marks]

This question concerns a `CarSales` program to manage the inventory of cars for a car dealer. The program stores the information about the cars in a field containing an array of `Car` objects. It also has a `count` field that contains the number of `Cars`, and the `Cars` are stored in cells 0 through `count-1` of the array. The program includes methods to list all the cars, add a car to the inventory, find cars under a given price, and remove all the cars of a given model.

Part of the `Car` class, for representing information about individual cars, is shown below.

---

```
public class Car{

    private int year;
    private String make;
    private String model;
    private int price;
    :

    public Car(Scanner in){
        make = in.next();
        model = in.next();
        :
    }

    public String toString(){
        return this.make + " " + this.model + " (" + this.year + ") $" + this.price;
    }
    public String getMake(){
        return this.make;
    }
    public String getModel(){
        return this.model;
    }
    public int getYear(){
        return this.year;
    }
    public int getPrice(){
        return this.price;
    }
    public void setPrice(int value){
        this.price = value;
    }
    :
}
```

**(Question 3 continued)**

The following are some of the fields and two of the methods of the `CarSales` class:

```
public class CarSales{

    private int maxCars = 200;
    private int count = 0;
    private Car [ ] inventory = new Car [maxCars];

    public void listCars(){
        for (int i=0; i<this.count; i++){
            UI.println (this.inventory[ i ].toString () );
        }
    }
    public boolean containsCar(Car car){
        for (int i=0; i<this.count; i++){
            if (this.inventory[ i ].equals(car) ){
                return true;
            }
        }
        return false;
    }
}
```

(a) [6 marks] Complete the following `addCar` method of the `CarSales` class. Its parameter is a `Car` and it should add the given `Car` to the inventory. If the array is full, the method should print out a message.

```
public void addCar (Car car){
```

```
}
```

(Question 3 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 3 continued)**

(b) [6 marks] Complete the following `findUnderPrice` method for the `CarSales` class. Its parameter is a price (an *int*), and it should print (to UI) all the cars in the inventory whose price is below the given price.

```
public void findUnderPrice(int price){
```

```
}
```

(c) [10 marks] Complete the following `deleteModel` method in the `CarSales` class. Its parameter is *String* specifying a model, and it should remove *all* cars of this model from the inventory. It is important that there should be no nulls in the inventory array in the range from 0 to `count-1`. The order the cars are stored in the inventory is not important and may be changed.

```
public void deleteModel(String model){
```

```
}
```

#### Question 4. Event Driven Input

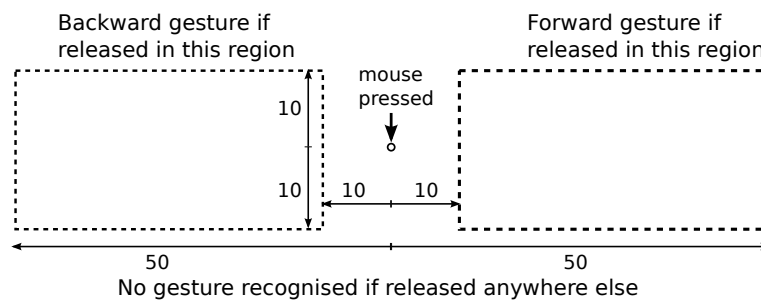
[20 marks]

(a) [8 marks] Some computer programs allow the user to give commands using “mouse gestures” — small strokes with the mouse. For example, a small stroke to the left might mean to move back a page, or moving in a small “U” shape might mean “undo”. Such programs have to be able to recognise different shapes.

You are to complete the following `TestGestures` program which recognises two simple gestures: a backward stroke and a forward stroke.

- The user can make a backward stroke by pressing the mouse at one point and releasing the mouse a bit to the left of that point.
- The user can make a forward stroke by pressing the mouse at one point and releasing the mouse a bit to the right of that point.

To count as an backward stroke, the `mouseReleased` point should be at least 10 pixels and at most 50 pixels to the left of the `mousePressed` point, and should not be more than 10 pixels above or below the `mousePressed` point. Forward strokes are similar. The diagram below shows where the `mouseReleased` point must be to be recognised as one of these gestures.



Complete the `TestGestures` program on the facing page so that it responds to forward and backward gestures. If the user makes a backward stroke, then the program should draw a red circle of diameter 60 at the point (10,10); if the user makes a forward stroke, then the program should draw a blue circle at the same point. If the user makes any other kind of gesture, then the program should draw a black circle.

(Question 4 continued on next page)

**(Question 4 continued)**

```
public class TestGestures implements MouseListener{
    private JFrame frame = new JFrame("TestGestures");
    private DrawingCanvas canvas = new DrawingCanvas();

    private int lastX;
    private int lastY;

    public TestGestures(){
        UI.setMouseListener(this);
    }

    public void mousePerformed(String action, double x, double y) {
        if (action.equals("pressed")){

        }
        else if (action.equals("released")){

        }
    }

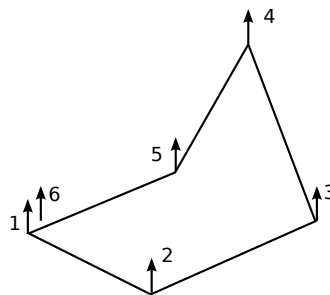
    UI.fillOval (10, 10, 60, 60);
}
}
```

(Question 4 continued on next page)

**(Question 4 continued)**

**(b)** [12 marks] (Harder) The MiniPaint program you wrote for assignment 6 had a very restricted set of shapes that the user could draw. A very common shape that drawing programs provide is a polygon, consisting of a sequence of straight line segments where the last line segment connects back to the first segment.

For example, to draw a polygon, a user might release the mouse at the sequence of locations (1 ... 6) shown in the figure below. The program would not draw anything on the first mouse release, but on subsequent mouse releases, it would draw a line from the new location to the previous location. When the user released the mouse at a location close to the first location (within 5 pixels of it), the program would draw a line from the first location (instead of the new location) and “finish” the polygon, ready to start a new one.



Complete the Polygon program on the facing page. You will only need to define fields and the mouseReleased method.

Hints:

- Your program must remember the first point and the most recent point, and must remember whether it is in the middle of drawing a polygon or not.
- mouseReleased must distinguish three situations: the first point of a polygon, the end of the polygon (releasing near the first point), and any other point of the polygon.
- You can compute the distance between  $(x_1, y_1)$  and  $(x_2, y_2)$  using `Math.hypot(x1-x2, y1-y2)`

(Question 4 continued on next page)

**(Question 4 continued)**

```
public class Polygon implements MouseListener{

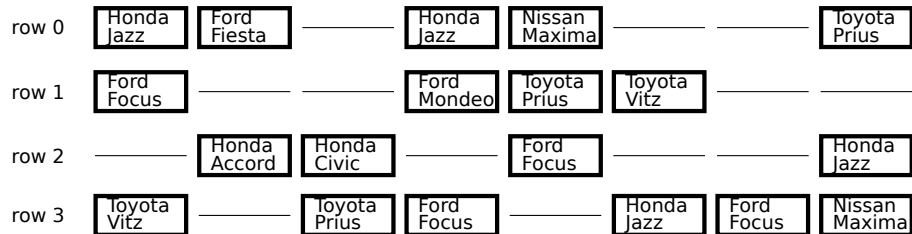
    public Polygon(){
        UI.addMouseListener(this);
    }
    public void mousePerformed(String action, double x, double y) {
        if (action.equals("released")){

        }
    }
}
```

## Question 5. 2D Arrays

[26 marks]

The CarYard program helps a car dealer manage the storage of cars in their car yard. The cars are lined up in rows, as in the following figure which shows a yard with 4 rows of cars:



The program has methods to help sales people find the location of a model in the yard, remove a car from the yard, count the number of cars in a row, and move cars around. The program uses a 2D array of Car objects and a null represents an empty space in a row. The Car class is described in question 3 on page 14, but this question is otherwise completely independent of question 3.

Field declarations and a constructor of the CarYard class are shown below:

```
public class CarYard{  
  
    private int rows;  
    private int rowLength;  
    private Car[ ][ ] layout;  
  
    public CarYard(int rs, int rl){  
        this.rows = rs;  
        this.rowLength = rl;  
        this.layout = new Car[rows][rowLength];  
    }  
}
```

(a) [6 marks] Complete the following removeCar method which should remove the car at a specified location (row and column) of the layout. If the location is already empty, it should print a message saying there is no car at the location.

```
public void removeCar(int row, int col){
```

```
}
```

(Question 5 continued on next page)

**(Question 5 continued)**

**(b)** [6 marks] Complete the following `countRow` method. Its parameter is the number of a row and it should count and return the number of cars in that row of the layout.

```
public int countRow(int row){
```

```
}
```

**(c)** [6 marks] Complete the following `findModel` method. Its parameter is a `String` specifying a model, and it should print (to UI) the location (row and column) of every car in the car yard that is of the specified model. For example, with the layout on the facing page, `findModel("Jazz")` should print out

```
Jazz at row 0, col 0  
Jazz at row 0, col 3  
Jazz at row 2, col 7  
Jazz at row 3, col 5
```

You will need to use method(s) from the `Car` class listed on page 14.

```
public void findModel(String model){
```

```
}
```

(Question 5 continued on next page)

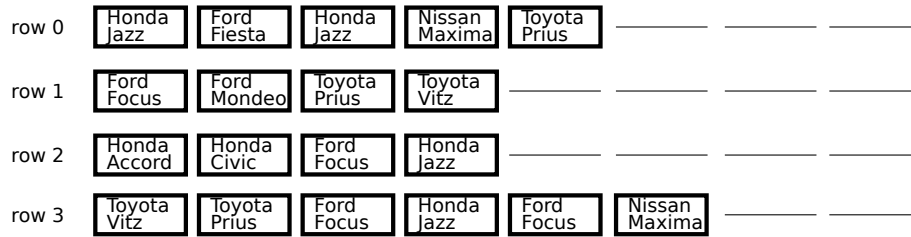
**SPARE PAGE FOR EXTRA ANSWERS**

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

**(Question 5 continued)**

**(d)** [8 marks] The sales people sometimes get rid of all the gaps in the yard by moving all the cars in each row as far to the front of the row as possible. Complete the following `closeUp` method which should move the cars in each row towards the front of the row, leaving all the spaces at the back of the row. The order of cars in each row should not change.

For example, calling `closeUp()` on the layout at the beginning of the question should result in the following layout:



```

public void closeUp(){

}
    
```

### Question 6. Designing with Interfaces

[10 marks]

The `CarSales` program in question 3 stored a collection of `Car` objects in an array to represent the inventory of cars in a car dealer. Suppose the car dealer wants to expand to have two more kinds of vehicles — `SUV`'s and `Vans` — in addition to cars. The three classes of vehicles have much in common; for example, they all have the methods `getMake`, `getModel`, `getPrice`, `setPrice`, `getYear` and `toString`, but some of the methods are implemented differently and the `SUV` and `Van` classes have some additional methods.

To extend the `CarSales` program to handle `SUV`s and `Vans`, you decide to create a `Vehicle` interface class, and make the `inventory` field of the `CarSales` class able to store `Cars`, `SUV`s, and `Vans`.

Complete the `Vehicle` interface class below, and make any necessary changes to the `CarSales` and the `Car` classes on the facing page (write your changes on the code in the boxes). The `listCars` and `containsCar` methods should work on all three kinds of vehicle.

You do **not** need to complete question 3 before this question.

```
public interface Vehicle {
```

```
}
```

## (Question 6 continued)

```

public class Car {
    private int year;
    private String make;
    private String model;
    private int price;
    :
    public Car(Scanner in){
        make = in.next();
        model = in.next ();
        :
    }
    public String toString(){
        return this.make + " " + this.model + " (" + this.year + ") $" + this.price;
    }
    public String getMake(){return this.make;}
    public String getModel(){return this.model;}
    public int getYear(){return this.year;}
    public int getPrice(){return this.price;}
    public void setPrice(int value){this.price = value;}
    :
}

```

```

public class CarSales{

    private int maxCars = 200;
    private int count = 0;
    private Car [ ] inventory = new Car [maxCars];

    public void listCars (){
        for ( int i=0; i<this.count; i++){
            Ul. println (this.inventory[i]. toString ());
        }
    }
    public boolean containsCar(Car car){
        for ( int i=0; i<this.count; i++){
            if (inventory[i]. equals(car) ){
                return true;
            }
        }
        return false;
    }
}

```

### Question 7. Debugging Loops

[12 marks]

The following `reverseValues` method is intended to reverse the order of the values in an array of Strings.

For example, given the array

words:	dog	bee	fox	gnu	cat	ant	eel
	0	1	2	3	4	5	6

it should change it to be

words:	eel	ant	cat	gnu	fox	bee	dog
	0	1	2	3	4	5	6

This version of `reverseValues` has multiple errors.

```
1 public void reverseValues(String[] array){
2     int j = 0;
3     int k = array.length;
4     while (j < array.length){
5         String temp = array[j];
6         array[k] = array[j];
7         array[j] = temp;
8         j = j+1;
9         k = k-1;
10    }
11 }
```

(a) [3 marks] Explain why one of the errors in `reverseValues` will cause the method to throw an exception (“crash”) and state the line on which the crash will happen.

(b) [3 marks] Circle and briefly describe at least one other error in `reverseValues`.

Answer on the code above

(Question 7 continued on next page)

**(Question 7 continued)**

**(c)** [6 marks] Write a correct version of `reverseValues` that does what it is supposed to do.

```
public void reverseValues(String[ ] array){
```

```
}
```

\*\*\*\*\*