



**EXAMINATIONS — 2006**

**END-OF-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.

There is documentation at the end of the paper.

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

**Questions**

	<b>Marks</b>
1. Understanding Java	[35]
2. Files	[12]
3. Programming with Loops	[15]
4. Implementing a Class	[15]
5. Arrays	[36]
6. Recursion	[30]
7. Inheritance	[37]

**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**

[35 marks]

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

```
int me = 2;
int us = me + 5;
int them;
if ( us > 7 )
    them = us * 3;
else
    them = us - 8;
System.out.printf("me=%d, us=%d, them=%d\n", me, us, them);
```

(b) [3 marks] What value will be returned by the call `basic("Hello")`?

```
public int basic(String str){
    if ( str.equals("hello") )
        return str.length();
    if ( str.equalsIgnoreCase("hello") )
        return str.length() * 2;
    return 15;
}
```

(c) [3 marks] What will the following code fragment print to `System.out`?

Note, the first line constructs an array of length 9 containing the specified strings.

```
String[] values = {"ant", "bee", "cat", "dog", "eel", "fox", "gnu", "hen", "imp"};
for( int k=0; k<values.length; k++ ){
    System.out.print(values[k]+ " ");
}
```

(Question 1 continued on next page)

**(Question 1 continued)**

**(d)** [5 marks] What will the following code fragment print to System.out?

```
String[] values = {"ant", "bee", "cat", "dog", "eel", "fox", "gnu", "hen", "imp"};
for ( int k=values.length-1; k>0; k=k/2 ){
    System.out.print(values[k]+ " ");
}
```

**(e)** [6 marks] Show the contents of the values array after the following code fragment has been run.

```
String[] values = {"ant", "bee", "cat", "dog", "eel", "fox", "gnu", "hen", "imp"};
for ( int k=1; k<values.length/2; k++){
    values[k] = values[values.length-k];
    values[values.length-k] = values[k];
}
```

0	1	2	3	4	5	6	7	8	

(Question 1 continued on next page)

**(Question 1 continued)**

(f) [5 marks] Suppose the file "output.log" contains the following text:

```
Log of errors from exam.pas
Line 015: warning: function "error" not yet defined
Line 018: error: variable "size" not declared
Line 025: warning: invalid syntax for "+"
Line 029: error: function "error" has no arguments
Line 038: error: file not closed
Line 048: warning: program has implicit END.
Compile Failed
```

What will the following method print out if it is called?

```
public void readLog(){
    try{
        int errors=0;
        Scanner sc = new Scanner (new File("output.log"));
        while ( sc.hasNext() ){
            String line = sc.nextLine();
            if ( line.contains("error") ){
                errors++;
                System.out.println( line.substring(5,8));
            }
        }
        sc.close();
        System.out.println(errors + " Errors");
    }
    catch(Exception e){System.out.println("Reading log file failed");}
}
```

(Question 1 continued on next page)

**(Question 1 continued)**

Consider the following Envelope class.

```
public class Envelope{
    // Fields
    private int size = 1;    // size must be 1, 2, or 3
    private int width;
    private double weight =0; //weight in grams
    private String address;

    // Constructor
    public Envelope(int sz){
        if (0 < sz && sz < 4)
            size = sz;
        System.out.println("New Envelope of size " + size);
    }

    // Methods
    public void setAddress(String s){
        if (address != null)
            System.out.println("Changing "+address);
        address = s;
    }

    public void addWeight(double w){
        weight = weight + w;
        if (weight > 100 * size){
            System.out.println("Envelope is now over-weight");
            address = address + "\n TOO HEAVY ";
        }
    }

    public String toString(){
        return ("Size "+ size + " envelope to "+address + " : "+weight +"gms");
    }
}
```

(Question 1 continued on next page)

**(Question 1 continued)**

**(g)** [10 marks] Given the *Envelope* class on the facing page, what will the following code fragment print out?

```
Envelope e = new Envelope(2);  
e.setAddress("Lindsay, VUW");  
e.addWeight(150);  
System.out.println(e.toString ());
```

```
System.out.println("-----");  
e.setAddress("Pondy, SMSCS");  
e.addWeight(80);  
System.out.println(e.toString ());
```

```
System.out.println("-----");  
e = new Envelope(4);  
e.setAddress("Marvin, SCPS");  
e.addWeight(110);  
System.out.println(e.toString ());
```

**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. Files**

[12 marks]

Complete the following `countFile` method so that it reads the contents of a file and prints out the number of lines and the number of characters in the file (not counting the end of line characters). The argument to `countFile` is the name of the file to read.

```
public void countFile(String fname){
    try {

        System.out.printf("Lines: %d\nCharacters: %d\n", lines, chars);

    }
    catch(Exception e){System.out.println("File reading failed");}
}
```

### Question 3. Programming with Loops

[15 marks]

(a) [5 marks] What will the following `printTable` method print out if it is called with an argument of 4?

```
public void printTable ( int n){
    for ( int k=0; k < n*n; k++){
        System.out.printf ("%2d ", k);
        if ( k==n )
            System.out.println ();
    }
}
```

(b) [10 marks] `printTable` was intended to print out an  $n \times n$  table of the integers starting at 1. The argument specifies how many rows and columns in the table. For example, if the argument is 3, `printTable` should print out:

```
1  2  3
4  5  6
7  8  9
```

If the argument is 5, it should print out:

```
1  2  3  4  5
6  7  8  9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```

On the facing page, write a correct version of `printTable` so that it does what it is supposed to.

(Question 3 continued on next page)



#### Question 4. Implementing a class

[15 marks]

The `DiaryPage` class is intended to represent pages from a diary or journal. A diary page should have a date and a title, and some text that is the entry for that day.

For example, a `DiaryPage` object with a date of the 3rd October 2006 and a title of “My Bicycle” and two lines of text might be printed out as follows:

```
date:  3/Oct/2006
title: My Bicycle
-----
Today, my bicycle broke.
When I stopped at the lights, the wheel fell off.
-----
```

Complete the `DiaryPage` class on the facing page.

The `DiaryPage` class should have fields to store a date, a title, and the text of the entry, a constructor, and four methods.

- The constructor should have three parameters: the day, month, and year, and should set the date of the page.
- `getDate()` should return the date in the form of a string. This method is already completed for you, and contains hints for representing the date.
- `setTitle(String t)` should set the title of the page. The title on a page can be set only once — `setTitle` should not change the title again once it has been set.
- `addToEntry(String txt)` should add a `String` to the current entry on the page. Each new string added to the entry should be start on a new line when it is printed out.
- `toString()` should construct a `String` representation of the page in the form shown in the example above, with the date, the title, and lines (made of `-`'s) before and after the entry.

The `DiaryPage` printed above could have been constructed with the calls

```
DiaryPage dp = new DiaryPage(3, "Oct", 2006);
dp.setTitle("My Bicycle");
dp.addToEntry("Today, my bicycle broke.");
dp.addToEntry("When I stopped at the lights, the wheel fell off.");
System.out.println(dp.toString ());
```

(Question 4 continued on next page)

**(Question 4 continued)**

```
/** Represents the information on a page of a diary */
public class DiaryPage{
    // fields

    /* Constructor */
    public DiaryPage(           ){

    }
    public String getDate(){
        return (day+" / "+month+" / "+year);
    }

    public void setTitle (String t){

    }

    public void addToEntry(String txt){

    }

    public String toString(){

    }

}
```

## Question 5. Arrays

[36 marks]

Suppose you own several copies of a very popular CD. You agree to lend copies to some of your friends, and you want to implement a program to keep track of who has borrowed copies of the CD. You are prepared to let people borrow more than one copy of the CD (one for home, one for the car, etc.), but you want to place a limit on the number that any person can borrow.

The following is an outline of a class called `Loans`, to be used as part of this program. This class records who has borrowed copies of your CD using an array of strings called `names`. The `Loans` constructor takes two integer arguments, the first is the number of copies of the CD that you own, and the second is the limit on the number of copies that any person can borrow.

The outline contains incomplete declarations for the constructor and four methods, which you are required to implement.

```
public class Loans {
    private String[] names;
    private int numNames;
    private int limit ;
    public Loans(int numCDs, int lim) { }
    public void printLoans() { }
    public void borrowItem(String who) { }
    public void returnItem(String who) { }
    public void printMultiLoans() { }
}
```

(a) [4 marks] Complete the `Loans` constructor so that it performs any necessary initialisation.

```
public Loans(int numCDs, int lim) {
}
}
```

(Question 5 continued on next page)

**(Question 5 continued)**

**(b)** [4 marks] Complete the `printLoans` method so that it prints a list of all the people who have borrowed your CD. If someone has borrowed more than one copy, their name should be printed once for each copy they have borrowed.

```
public void printLoans() {
```

```
}
```

**(c)** [8 marks] Complete the `borrowItem` method so that it records that `who` has borrowed a copy of your CD, by adding his/her name to `names` — provided that he/she has not already borrowed the maximum number allowed. Print a suitable message if `who` has already reached the borrowing limit.

```
public void borrowItem(String who) {
```

```
}
```

(Question 5 continued on next page)

**(Question 5 continued)**

**(d)** [8 marks] Complete the `returnItem` method so that it records that `who` has returned a copy of your CD, by removing one occurrence of his/her name from `names`. Print a suitable message if `who` does not currently have any copies of your CD.

```
public void returnItem(String who) {
```

```
}
```

(Question 5 continued on next page)

**(Question 5 continued)**

(e) [12 marks] Complete the `printMultiLoans` method so that it prints a list of all the people who currently have two or more copies of your CD, along with the number of copies that each such person has.

To get full marks, you should ensure that no person is printed more than once in this list. You may define additional methods.

```
public void printMultiLoans() {
```

```
}
```

## Question 6. Recursion

[30 marks]

(a) [6 marks] Consider the following recursive method.

```
public int fn( int x, int y){
    if ( x == 1 )
        return y;
    else
        return fn(x/2, y+1);
}
```

For each of the following calls, show the sequence of calls performed and the final value that will be returned.

(i) [2 marks] `fn(1, 0);`

(ii) [2 marks] `fn(2, 0);`

(Question 6 continued on next page)

**(Question 6 continued)**

**(iii)** [2 marks] `fn(13, 0);`

**(b)** [8 marks] Write an equivalent version of `fn` in part **(a)**, using a loop instead of recursion.

```
public int fn(int x, int y) {
```

```
}
```

(Question 6 continued on next page)

**(Question 6 continued)**

(c) [6 marks] Consider the following recursive method.

```
public void rec(String str) {  
    int n = str.length();  
    if ( n <= 2 ) {  
        System.out.print(str);  
    } else {  
        rec(str.substring(n/2, n));  
        rec(str.substring(0, n/2));  
    }  
}
```

For each of the following calls, show the sequence of calls performed and the output that is produced.

(i) [3 marks] `rec("abcd")`;

(ii) [3 marks] `rec("recursion")`;

(Question 6 continued on next page)

**(Question 6 continued)**

**(d)** [10 marks] Consider the following method.

```
public boolean pal(int[] a) {
    int i = 0;
    int j = a.length-1;
    while ( i < j ) {
        if ( a[i] == a[j] ) {
            i++;
            j--;
        } else {
            return false;
        }
    }
    return true;
}
```

Write an equivalent `pal` method, using recursion instead of a loop.

**Hint:** You will need to define a recursive helper method.

**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 7. Inheritance**

[37 marks]

(a) [8 marks]

Consider the following code for two classes, `Adder` and `Badder`, and the tester method which uses these classes.

```
public class Adder {
    protected int x = 2;
    public Adder() {
        display ();
    }
    public void inc() {
        x++;
        display ();
    }
    public void dec() {
        x--;
        display ();
    }
    protected void display() {
        System.out.print(x + " ");
    }
}

public class Badder extends Adder {
    public void dec() {
        x=x-2;
        display ();
    }
}
```

```
public static void tester () {

    Adder a = new Adder();
    a.inc ();
    a.dec();
    a.inc ();
    a.dec();
    System.out.println ();

    Adder b = new Badder();
    b.inc ();
    b.dec();
    b.inc ();
    b.dec();
    System.out.println ();

}
```

What will `tester` print out when it is executed?

(Question 7 continued on next page)

**(Question 7 continued)**

**(b)** [11 marks]

Consider the following class declarations:

```
class A {  
  
    public String eats() {  
        return "anything";  
    }  
  
    public String says() {  
        return "nothing";  
    }  
}  
  
class B extends A {  
  
    public String eats() {  
        return "grass";  
    }  
}  
  
class C extends A {  
  
    public String says() {  
        return "woof";  
    }  
}  
  
class D extends B {  
  
    public String says() {  
        return "baa";  
    }  
}
```

(Question 7 continued on next page)

**(Question 7 continued)**

Show the output that will be produced by each of the following code fragments.

**(i)** [2 marks]

```
A animal = new A();  
System.out.println("Eats " + animal.eats() + " and says " + animal.says());
```

**(ii)** [3 marks]

```
A animal = new B();  
System.out.println("Eats " + animal.eats() + " and says " + animal.says());
```

**(iii)** [3 marks]

```
A animal = new C();  
System.out.println("Eats " + animal.eats() + " and says " + animal.says());
```

**(iv)** [3 marks]

```
A animal = new D();  
System.out.println("Eats " + animal.eats() + " and says " + animal.says());
```

(Question 7 continued on next page)

**(Question 7 continued)**

**(c)** [18 marks]

A computer system for a carrier company uses a class called `CarrierOrder` to store information about orders for goods to be transported and to compute the cost for such orders. As shown below, the class includes a constructor which takes as arguments the volume and weight of the goods to be transported and the distance they are to be taken. It also includes a method called `cost` which computes the cost for the order, and a method called `printOrder` which prints details of the order.

```
class CarrierOrder {  
  
    public CarrierOrder(double vol, double wgt, double dist) {  
    }  
  
    public double cost() {  
    }  
  
    public void printOrder() {  
    }  
  
    ...  
}
```

The carrier company decides to offer an express service, for which they guarantee same day delivery. This service is only available for orders upto a specified maximum volume, weight and distance, and the price is 50% greater than the normal price for the same order. Express orders that do exceed the maximum volume, weight or distance are treated like ordinary orders and charged at the normal rate.

Express orders will be handled by a new `ExpressOrder` class which should be a subclass of `CarrierOrder`. An outline of the `ExpressOrder` class is shown opposite. The maximum volume, weight and distance for an express order are specified in `static` fields.

You are required to implement the `ExpressOrder` constructor, and make whatever other additions are required so that the `cost` and `printOrder` methods work correctly for `ExpressOrder` objects.

The details printed by `printOrder` should be exactly the same as is printed for a standard order.

You must not modify `CarrierOrder` in any way. You do not need to know what fields `CarrierOrder` has, how these methods are defined, or what other methods `CarrierOrder` has, except that they cannot change the volume, weight or distance of the order.

(Question 7 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.

## Brief and partial documentation of some classes and methods

**PrintStream class:**

```
public PrintStream (File f); // Note, System.out is a PrintStream object
public void close (); // Constructor, for printing to a file
public void print (String s); // Close the file (if it is wrapping a File object)
public void print (int i); // Prints s with no newline
public void print (double d); // Prints i with no newline
public void println (); // Prints d with no newline
public void println (String s); // Prints a newline
public void println (int i); // Prints s followed by newline
public void println (double d); // Prints i followed by newline
public void printf (String format, ...); // Prints d followed by newline
// Prints the format string, inserting the remaining
// arguments at the %'s in the format string:
// %s for Strings.
// %d for ints, (%3d: use at least 3 characters),
// %.2f for 2dp doubles,
// (%6.2f: at least 6 characters and 2dp),
// Use \n for newline
```

**Scanner class:**

```
public Scanner (InputStream i); // Constructor. Note: System.in is an InputStream
public Scanner (File f); // Constructor, for reading from a file
public Scanner (String s); // Constructor, for reading from a string
public boolean hasNext(); // Returns true if there is more to read
public boolean hasNextInt(); // Returns true if the next token is an integer
public boolean hasNextDouble(); // Returns true if the next token is a number
public String next(); // Returns the next token (chars up to a space/line)
public String nextLine(); // Returns the next line
public int nextInt (); // Returns the integer value of the next token
// (throws exception if next token is not an integer)
public double nextDouble(); // Returns the double value of the next token
// (throws exception if next token is not a number)
public void close (); // Closes the file (if it is wrapping a File object)
```

**File class:**

```
public File (String fname); // Constructor. Creates a File object attached to the
// file with the name fname
```

**String class:**

```
public int length (); // Returns the length (number of characters) of the string
public boolean equals (String s); // String has same characters as s
public boolean equalsIgnoreCase (String s); // String has same characters as s, ignoring their case
public boolean startsWith (String s); // First part of string matches s
public boolean contains (String s); // s matches some part of the string
public String substring (int j, int k) // Returns substring from index j to index k-1
public int indexOf (String s); // Returns -1 if it does not contain s anywhere
// otherwise, returns the index of where s first matches
```

*Integer class:*

```
public static final int MAX_VALUE; // The largest possible int:  $2^{(31-1)}$   
public static final int MIN_VALUE; // The smallest possible int:  $-2^{(31)}$ 
```

*Double class:*

```
public static final double MAX_VALUE; // The largest possible double: just under  $2^{(1024)}$   
public static final double MIN_VALUE; // The smallest possible positive nonzero double  
public static final double POSITIVE_INFINITY; // positive infinity (greater than any number)  
public static final double NEGATIVE_INFINITY; // negative infinity (less than any number)  
public static final double NaN; // The Double that is Not a Number ("undefined")
```

*Math class:*

```
public static double sqrt(double x); // Returns the square root of x  
public static double min(double x, double y); // Returns the smaller of x and y  
public static double max(double x, double y); // Returns the larger of x and y  
public static double abs(double x); // Returns the absolute value of x  
public static int min(int x, int y); // Returns the smaller of x and y  
public static int max(int x, int y); // Returns the larger of x and y  
public static int abs(int x); // Returns the absolute value of x
```

*DrawingCanvas class:*

```
public void clear(); // Clears the drawing canvas  
public void setForeground(Color c); // Change the colour for later commands  
public void drawLine(int x, int y, int u, int v); // Draws line from  $cd\{x, y\}$  to  $cd\{u, v\}$   
public void drawRect(int x, int y, int wd, int ht); // Draws outline of rectangle  
public void fillRect(int x, int y, int wd, int ht); // Draws solid rectangle  
public void clearRect(int x, int y, int wd, int ht); // Draws clear rectangle  
public void drawOval(int x, int y, int wd, int ht); // Draws outline of oval  
public void fillOval(int x, int y, int wd, int ht); // Draws solid oval
```

*Color class:*

```
public Color(int red, int green, int blue); // Make a colour; arguments must be 0..255  
Color.gray, Color.blue, Color.red, // Some of the predefined colours  
Color.green, Color.black, Color.white
```