Student ID: .....

EXAMINATIONS — 2006

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.

There is documentation at the end of the paper.

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

Questions

	Marks
1. Basic Java	[58]
2. Arrays and Loops	[20]
3. Files and Loops	[12]
4. Arrays of objects	[30]
5. Buttons and Event Driven Input	[10]
6. Recursion	[25]
7. Inheritance	[25]

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. Basic Java [58 marks]

(a) [6 marks] What will the following simple method print out?

```
public void simple(){
   int n = 4;
   System.out.println(n + " : " + n*n);
   n = n + 2;
   int y = n * 2 / 5;
   System.out.printf("n = %d, y = %d\n", n, y);
   System.out.printf("Name %s tag\n", "John");
}
```

```
4 : 16
n = 6, y = 2
Name John tag
```

(Question 1 continued on next page)

(Question 1 continued)

Consider the following code fragment which uses a variable x of type *int*.

```
if ( x < 10 ) {
   System.out.printf("%d is black\n", x);
   x = x * 10;
}
if ( x < 50 ) {
   System.out.printf("%d is red\n", x);
}
else if ( x < 90 ) {
   System.out.printf("%d is blue\n", x);
}
else {
   System.out.printf("%d is green\n", x);
}
```

(b) [3 marks] What will the fragment print out if x initially contains the value 4?

4 is black
40 is red

(c) [3 marks] What will the fragment print out if x initially contains the value 6?

6 is black
60 is blue

(d) [2 marks] What will the fragment print out if x initially contains the value 16?

16 is red

(Question 1 continued on next page)

(Question 1 continued)

(e) [4 marks] What will the following code fragment print out?

```
for (int num = 1; num < 40; num = num * 2){
   System.out.println(num);
}
```

```
1
2
4
8
16
32
```

(f) [6 marks] Write a fragment of code that will print out a table of the square root of each integer from 50 to 100, inclusive. Each line should contain an integer and its square root printed to four decimal places. For example, the first line of the output should be:

```
50   7.0711
```

Hint: See the documentation at the end of the exam paper for the method in the Math class for computing the square root of a number.

```
for (int n = 50; n <= 100; n++){
   System.out.printf(" %3d  %7.4f", n, Math.sqrt(n));
}
```

(Question 1 continued on next page)

(Question 1 continued)

(g) [5 marks] Write a method called isNull which takes a parameter of type *String* and returns true if its argument is **null**, and returns false otherwise.

```
public... | boolean isNull(String s){
|    return (s == null);
|    // OR
|    if ( s == null )  return true;
|    else   return false;
|  }
```

(h) [6 marks] Complete the following compareStrings method so that it returns -1 if str1 is shorter than str2; returns 0 if the two strings are the same length; and returns 1 if str1 is longer than str2. You may assume that str1 and str2 are not null.

Hint: the length() method returns the length of a *String*.

```
public int compareStrings(String str1, String str2){
|    if ( str1.length()  <  str2.length() )
|       return –1;
|    else if ( str1.length()  ==  str2.length() )
|       return 0;
|    else
|       return 1;
|
|  // Note that the else's are unnecessary.
|
}
```

(Question 1 continued on next page)

(Question 1 continued)

(i) [5 marks] Consider the following printWords method.

```
public void printWords(String[ ] wds){
    int i = 0;
    while ( i < wds.length ){
        System.out.printf("%d : %s\n", i, wds[i]);
        i = i + 2;
    }
}
```

Suppose that printWords is called with the following array as its argument. What will it print out?

"ant"	"bee"	"cat"	"dog"	"eel"	"fox"	"gnu"	"hen"	"jay"	"kea"
0	1	2	3	4	5	6	7	8	9

```
0 : ant
2 : cat
4 : eel
6 : gnu
8 : jay
```

(j) [5 marks] Complete the following writeWords method so that it prints all the words in the wds array in reverse order (from the last word to the first word), one word per line. You may assume that there are no **null** values in the array.

```
public void writeWords(String[ ] wds){
    for (int i = wds.length–1; i >= 0; i--)
        System.out.println(wds[i]);
    // OR
    int j = wds.length–1;
    while ( j >= 0 ){
        System.out.println(wds[j]);
        j = j–1;
    }
}
```

(Question 1 continued on next page)

(Question 1 continued)

Consider the following checkFile method.

```
public void checkFile(String fname){
   try{
      Scanner sc  =  new Scanner(new File(fname));
      if ( !sc.hasNext() )
         System.out.println("File check A");
      else {
         System.out.println(sc.next());
         if ( sc.hasNext() )
            System.out.println("File check B");
         else
            System.out.println("File check C");
      }
   }
   catch(Exception e){}
}
```

(k) [3 marks] Suppose checkFile were called with the name of a file that contained the following text, what would it print out?

```
this is a simple file
```

this
File check B

(l) [3 marks] Suppose checkFile were called with the name of a file that contained the following text, what would it print out?

```
Quit
```

Quit
File check C

(Question 1 continued on next page)

(Question 1 continued)

(m) [7 marks] Complete the printFile method below so that it reads a file and prints (to System.out) each word of the file on a separate line.

For example, the file shown below on the left should be printed out as shown on the right:

File

```
A file with several
words on each line
```

Output

```
A
file
with
several
words
on
each
line
```

```
public void printFile(){
   try{
      String fileName = FileDialog.open();
      Scanner sc = new Scanner(new File(fileName));
      while ( sc.hasNext() ){
         System.out.println(sc.next());
      }
      sc.close();




   }
   catch(Exception e){}
}
```

Question 2. Arrays and Loops [20 marks]

Consider the following q2 method:

```
public int q2(int[ ] data){
   int ans  =  0;
   for (int i = 0; i < data.length; i++){
      if ( data[i] == 0 )
         ans++;
   }
   return ans;
}
```

(a) [3 marks] What value would q2 return if called with the following array as its argument?

5	1	0	4	1	3	0	4	5	0
0	1	2	3	4	5	6	7	8	9

3

(b) [2 marks] Write a description of what the q2 method computes.

It computes the number of zeros in the array

(Question 2 continued on next page)

(Question 2 continued)

(c) [15 marks] The following largeValues method is passed an array of integers. Complete largeValues so that it prints out the largest and the second largest value in the array. You may assume that the length of the array is at least two, and that all the numbers are different.

Note: You can get partial marks for code that just prints the largest value in the array.

```
public void largeValues(int [ ] data){
    int largest = Math.max(data[0], data[1]);
    int second = Math.min(data[0], data[1]);
    for (int i = 2; i < data.length; i++){
        if ( data[i] > largest ){
            second = largest;
            largest = data[i];
        }
        else if ( data[i] > second ){
            second = data[i];
        }
    }
    System.out.printf("Largest: %d\nSecond: %d\n", largest,second);
  OR (not such a nice design)
    int largest = data[0];
    for (int i = 1; i < data.length; i++){
        if (data[i] > largest)
            largest = data[i];
    }
    int second = data[0];
    if (second == largest) second = data[1];
    for (int i = 1; i < data.length; i++){
        if (data[i] > second && data[i] < largest)
             second = data[i];
    }
    System.out.printf("Largest: %d\nSecond: %d\n", largest,second);
}
```

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. Files and Loops [12 marks]

The compareFiles method below takes two file names as arguments. Complete compareFiles so that it reads the two files, comparing them line by line, and prints out any lines in the first file that are different from the corresponding line in the second file. If the first file has more lines than the second file, then it should also print out the extra lines of the first file. It should ignore any extra lines in the second file. Every line printed should have its line number printed at the beginning.

For example, given File1 and File2 shown on the left below, compareFiles should produce the output shown on the right.

File 1

```
This file has
many words
on lines
(and punctuation!)
but no numbers.
```

File 2

```
This file has
some words
on lines
```

Output

```
2: many words
4: (and punctuation!)
5: but no numbers.
```

Hint: You will need two Scanner objects.
The nextLine() method in the Scanner class will return the next line from a Scanner.

```
public void compareFiles(String fn1, String fn2){
  try{
    Scanner sc1 = new Scanner(new File(fn1));
    Scanner sc2 = new Scanner(new File(fn2));
    int ln = 0;
    while ( sc1.hasNext() ){
      String line = sc1.nextLine();
      ln++;
      if ( !sc2.hasNext() || ! line.equals(sc2.nextLine()) ){
        System.out.printf("%d : %s\n", ln, line);
      }
    }
    sc1.close();
    sc2.close();



  }
  catch(Exception e){}
}
```

Question 4. Arrays of Objects [30 marks]

A Large Building Consent Form consists of a number of sections, usually written by different people. Each section is a separate document, and the documents will be revised and updated frequently during the writing process. The ConsentForm and Document classes below are part of a program to help a manager to keep track of the latest version of each section of a consent form.

A Document object contains the author, date, filename, and revision number of a document. The Document class contains several methods for accessing and setting these fields. The headers of these methods are shown below. You will need to use these methods in the rest of the question, but you do not need to know how they are defined.

```
public class Document{

 /** Constructor */
 public Document(String author, Date date, String filename){...}

 /** Returns the Date of the document */
 public Date getDate(){...}

 /** Returns true if and only if the date of this document is more recent
     than the specified date */
 public Boolean newer(Date d){...}

 /** Returns the revision number of the document */
 public int getRevisionNumber(){...}

 /** Sets the revision number of the document */
 public  void setRevisionNumber(int n){...}

 /** Returns a short description of the document */
 public String getShortDescription(){...}
}
```

(Question 4 continued on next page)

(Question 4 continued)

A ConsentForm object contains an array of Document objects. Each element of the array contains the document for one section of the form. The sections are numbered from 0. The ConsentForm class has three methods for listing the current documents in the form, updating the document in a particular section, and for removing all documents that are older than a specified date. You are to complete these three methods. The fields and the constructor of the ConsentForm class are shown below.

```
public class ConsentForm{

 private String title;              // Title of the consent form
 private Document[ ] sections; // Array of Documents for each section, numbered from 0
                                    // A section will be null if no Document has been added yet.

 /** Creates a new consent form given a title and a number of sections.
    Constructs an empty array with the correct number of sections. */
 public ConsentForm(String t, int numSections){
  title = t;
  sections = new Document[numSections];
 }

  ...
```

(a) [10 marks] Complete the following listSections method so that it prints out the current status of the consent form.

listSections should first print the title. Then, for each section, it should print the number of the section followed either by the short description of the current document for that section, or by `"No document"` if there is no document yet for that section.

```
public void listSections(){
   System.out.printf("Title: %s\n", title);
   for (int sn = 0; sn < this.sections.length; sn++){
      if ( this.sections[sn] == null )
         System.out.printf("%2d: No document\n", sn);
      else
       System.out.printf("%2d: rev: %d\n", sn, this.sections[sn].getDescription());
   }
}
```

(Question 4 continued on next page)

(Question 4 continued)

(b) [8 marks] Complete the following removeOld method so that it removes all documents that were written on or before the date in the argument.

```
public void removeOld(Date date){
    for (int sn = 0; sn < this.sections.length; sn++){
        if ( this.sections[sn] != null  &&  ! this.sections[sn].newer(date) )
            this.sections[sn] = null;
    }




}
```

(Question 4 continued on next page)

(Question 4 continued)

(c) [12 marks] The updateSection method takes as parameters the number of a section to be updated and a new Document for that section. Complete the updateSection method below so that it attempts to update the specified section as follows:

- If there is no current document for the specified section, then updateSection should set the revision number of the new document to 1 and store the new document in the appropriate place in the array.
- If the current document for the specified section is newer than the new document, then updateSection should not replace the current document.
- Otherwise, updateSection should set the revision number of the new document to 1 more than the revision number of the current document, and replace the current document by the new document.

updateSection should return true if it changed the ConsentForm to contain the new document, and false otherwise.

```
public boolean updateSection(int sectNum, Document newDoc ){
    Documents curDoc = this.sections[sectNum];
    if ( curDoc  ==  null ){
       newDoc.setRevisionNumber(1);
       this.sections[sectNum]  =  newDoc;
       return true;
    }
   else if ( curDoc.newer(newDoc.getDate()) )
       return false;
   else {
       newDoc.setRevisionNumber(curDoc.getRevisionNumber() +1);
       this.sections[sectNum] =  newDoc;
       return true;
    }

}
```

Question 5. Buttons and Event Driven Input [10 marks]

The SimpleGui program on the facing page has a simple GUI, in the same style as programs for several assignments in the course. The GUI contains two buttons and a DrawingCanvas. If the user presses the *Square* button, the program should clear the canvas and draw a filled red square at the point (100, 100). If the user presses the Circle button, the program should clear the canvas and draw a filled green circle at the same point. Both shapes should be of size 10.

The GUI is set up for you; you are to complete the actionPerformed method which responds to button events.

Hints:

- getActionCommand() is a method of the ActionEvent class. It returns the name of the button that was clicked.
- See the documentation at the end of the exam paper for methods of the DrawingCanvas class.
- Color.red and Color.green are constants representing the colours red and green.

(a) [4 marks] Sketch the user interface of the SimpleGui program.

(Question 5 continued on next page)

(Question 5 continued)

(b) [6 marks] Complete the actionPerformed method below.

```
public class SimpleGui implements ActionListener{
   private JFrame frame;
   private DrawingCanvas canvas;

   public SimpleGui(){
      frame  =  new JFrame("SimpleGui");
      frame.setSize(600, 400);

      canvas  =  new DrawingCanvas();
      frame.getContentPane().add(canvas, BorderLayout.CENTER);

      JPanel panel  =  new JPanel();
      frame.getContentPane().add(panel, BorderLayout.NORTH);

      JButton circleButton  =  new JButton("Circle");
      circleButton.addActionListener(this);
      panel.add(circleButton);

      JButton squareButton  =  new JButton("Square");
      squareButton.addActionListener(this);
      panel.add(squareButton);

      frame.setVisible(true);
   }

   public void actionPerformed(ActionEvent e){

      if ( e.getActionCommand().equals("Circle") ){
          canvas.clear();
          canvas.setForeground(Color.green);
          canvas.fillOval(100,100, 10, 10);
      }
      else if ( e.getActionCommand().equals("Square") ){
          canvas.clear();
          canvas.setForeground(Color.red);
          canvas.fillRect(100,100, 10, 10);
      }



   }
}
```

Question 6. Recursion [25 marks]

(a) [7 marks] Consider the following recursive method:

```
public void Rec(int n) {
   if ( n <= 1 )
      System.out.println(n);
   else {
      System.out.println(n);
      Rec(n–1);
      System.out.println(n);
   }
}
```

Show the output that will be produced for each of the following calls:

(i) [1 mark] Rec(1)

```
1
```

(ii) [3 marks] Rec(2)

```
2 1 2
```

(iii) [3 marks] Rec(4)

```
4 3 2 1 2 3 4
```

(Question 6 continued on next page)

(Question 6 continued)

(b) [8 marks] The mathematical function B is defined for positive integers as follows:

$$\begin{aligned} B(1,n) &= 1, \text{ for } n \geq 1 \\ B(m,1) &= 1, \text{ for } m \geq 1 \\ B(m,n) &= B(m-1,n) + B(m,n-1), \text{ for } m,n > 1 \end{aligned}$$

Write a method to compute B for any positive integers m and n.

```
public int B(int m, int n) {
   if ( m == 1 || n == 1 )
      return 1;
   else
      return B(m–1, n) + B(m, n–1);
}
```

(Question 6 continued on next page)

(Question 6 continued)

(c) [10 marks] Consider the following method, count, which is implemented using a recursive helper method, also called count:

```
public int count(int[ ] data) {
   return count(data, 1, 0);
}

public int count(int[ ] data, int k, int n) {
   if ( k == data.length )
      return n;
   else if ( data[k] < data[k–1] )
      return count(data, k+1, n+1);
   else
      return count(data, k+1, n);
}
```

Write an equivalent version of count which uses a loop instead of using recursion.

```
public int count(int[ ] data) {
   int n = 0;
   for (int k = 1; k < data.length; k++)
      if ( data[k] < data[k–1] )
         n++;
   return(n);
}
```

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 [25 marks]

(a) [8 marks]

Consider the following code for two classes, A and B, and the testAB method which uses these classes.

```
public class A {

   protected int x = 1;

   public A() {
      display();
   }

   public void inc() {
      x++;
      display();
   }

   public void dec() {
      x--;
      display();
   }

   protected void display() {
      System.out.print(x + " ");
   }
}
```

```
public class B extends A {

   public void inc() {
      x = x*2;
      display();
   }
}
```

(Question 7 continued on next page)

(Question 7 continued)

```
public static void testAB() {

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

    A b = new B();
    b.inc();
    b.inc();
    b.dec();
    b.inc();
    System.out.println();

}
```

What would testAB print out?

```
1 2 3 2 3
1 2 4 3 6
```

(Question 7 continued on next page)

(Question 7 continued)

(b) [17 marks]

A sales system for a shop uses a class called StockItem to store information about the items the shop sells and to compute the price for a given quantity of an item. The class includes a number of other methods which are not shown here.

```
public class StockItem {

   private String name;
   private double unitPrice;

   public StockItem(String n, double p) {
      name = n;
      unitPrice = p;
   }
   public double price(int qty) {
      return unitPrice*qty;
   }

      ⋮
}
```

The shop owner occasionally offers a discount on certain items when at least some minimum quantity are bought. You are required to write a class called DiscountItem to support this.

The constructor for DiscountItem should take as arguments the name, unit price, minimum number and discount (as a percentage). Its price method should return the standard price (as computed by the price method in StockItem) if the number bought is less than the minimum, and should otherwise reduce the price by the given percentage. All other methods defined in StockItem should work in exactly the same way for DiscountItem objects.

Your DiscountItem should be a subclass of StockItem and should utilise features of StockItem as much as possible.

(Question 7 continued on next page)

(Question 7 continued)

```
public class DiscountItem extends StockItem {

    private int min;
    private float discount;

    public DiscountItem(String n, double p, int m, float d){
        super(n, p);
        min = m;
        discount = d;
    }

    public double price(int qty){
        double p = super.price(qty);
        if ( qty >= min )
            p = p–p∗discount/100;
        return p;
    }
  ...
}
```

****

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:

Note, System.out *is a* PrintStream *object*

public PrintStream(*File* f);	*Constructor, for printing to a file*
public *void* close();	*Close the file (if it is wrapping a File object)*
public *void* print(*String* s);	*Prints* s *with no newline*
public *void* print(*int* i);	*Prints* i *with no newline*
public *void* print(*double* d);	*Prints* d *with no newline*
public *void* println();	*Prints a newline*
public *void* println(*String* s);	*Prints* s *followed by newline*
public *void* println(*int* i);	*Prints* i *followed by newline*
public *void* println(*double* d);	*Prints* d *followed by newline*
public *void* printf(*String* format, ...);	*Prints the* format *string, inserting the remaining arguments at the* %*'s in the format string:* %3d *for* int*s, (using at least 3 characters),* %4.2f *for* double*s (4 characters and 2 decimal places),* %s *for* String*s.* *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 *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

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})
public static *int* parseInt(*String* str);	*Returns the integer represented by the string*

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 *int* indexOf(*String* s);	*Returns -1 if it does not contain* s *anywhere otherwise, returns the index of where* s *first matches*

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* (x, y) *to* (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*