



EXAMINATIONS — 2007

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, which you may tear off.

This includes example programs showing a variety of Java syntax.

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

Questions

	Marks
1. Understanding Java	[60]
2. Files	[20]
3. Programming with Loops	[20]
4. Arrays of Objects	[27]
5. Designing with Interfaces	[30]
6. Recursion	[23]

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

[60 marks]

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

```
double a = 2.5;
int b = 3;
String expr = a + "!=" + b;
a = a * b;
b = b / 2;
System.out.printf("a= %4.2f b= %d expr= %s\n", a, b, (expr + "=true"));
```

```
| a= 7.50 b= 1 expr= 2.5!=3=true
```

(b) [6 marks] Consider the following compute method:

```
public String compute(int x, String b) {
    if (x < 3 || b.length() < 2)
        return "too small";
    else if (x >= 5 && b.length() <= 4)
        return "yes: " + b;
    else if (b.length() > x)
        return x + "--" + b;
    else
        return "all true";
}
```

What would the following calls to compute return?

```
compute(5, "YY") ==> | yes: YY
compute(4, "x") ==> | too small
compute(4, "four") ==> | all true
compute(3, "three") ==> | 3 - - three
```

(Question 1 continued on next page)

(Question 1 continued)

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

```
for (int k = 8; k > 0; k=k-2){
    System.out.println("k = " + k);
}
System.out.println("Done");
```

```
k = 8
k = 6
k = 4
k = 2
Done
```

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

```
String message = "hu";
while (true){
    System.out.printf("num = %d, " + message.length());
    message = message + "loa";
    if (message.length() > 11)
        break;
    System.out.println("msg: " + message);
}
System.out.printf("Final: %s\n", message);
```

```
num = 2, msg:  huloa
num = 5, msg:  huloaloa
num = 8, msg:  huloaloaloa
num = 11, Final:  huloaloaloaloa
```

(Question 1 continued on next page)

(Question 1 continued)

(e) [5 marks] Suppose the variable `words` is declared to be an array containing 8 strings:

```
String [ ] words = {"bee","cow","ant","car","cat","bat","ape","can"};
```

words:	bee	cow	ant	car	cat	bat	ape	can
	0	1	2	3	4	5	6	7

What will the following code fragment print out?

```
for ( int index=words.length-1; index > 0; index-- ){
    System.out.print(index + ": " + words[index]+ ", ");
}
```

```
7: can, 6: ape, 5: bat, 4: cat, 3: car, 2: ant, 1:
cow
```

(f) [6 marks] Suppose that the variable `words` is declared as before:

```
String [ ] words = {"bee","cow","ant","car","cat","bat","ape","can"};
```

Show the contents of `words` after the following `exchange` method is called with the arguments `exchange(words, 3)`.

```
public void exchange(String [ ] wds, int limit ){
    for( int i = 0; i < limit; i++){
        int k = limit + i;
        wds[k] = wds[i];
        wds[i] = wds[k];
    }
}
```

words:	bee	cow	ant	bee	cow	ant	ape	can
	0	1	2	3	4	5	6	7

(g) [3 marks] For what values of `n` would `exchange(words, n)` result in an error?

```
any value of n greater than 4 will cause an error
since it will attempt access words[limit + (limit-1)]
```

(Question 1 continued on next page)

(Question 1 continued)

The `Bouncer` class on the facing page defines `Bouncer` objects, which have two fields to store their weight and path. The class also contains a `test` method.

(h) [5 marks] If the `test` method is called, what will it print out?

```
A: 3.5 kg along a ZigZag
B: 2.1 kg along a ZigZag
C: 7.0 kg along a Curve
D: 7.0 kg along a Curve
E: 8.4 kg along a Spiral
```

(i) [5 marks] Write an additional method called `unstable` for the `Bouncer` class which returns `true` if the weight of the `Bouncer` is above 4.0 and the path is a `Curve` or a `Spiral`. It should return `false` in all other cases.

```
public boolean unstable(){
    return (this.weight > 4.0 && (this.path.equals("Curve") ||
        this.path.equals("Spiral")));
}
```

(Question 1 continued on next page)

(Question 1 continued)

```
public class Bouncer{
    private double weight;
    private String path;

    public Bouncer(double z){
        this.weight = z;
        this.path = "ZigZag";
    }

    public void changePath(String n){
        this.weight = this.weight * 2;
        this.path = n;
    }

    public String toString(){
        return (this.weight + " kg along a " +this.path);
    }

    public static void test(){
        Bouncer b1 = new Bouncer(3.5);
        Bouncer b5 = new Bouncer(2.1);

        System.out.println("A: " + b1.toString());
        System.out.println("B: " + b5.toString());

        b1.changePath("Curve");
        b5.changePath("Straight line");
        System.out.println("C: " + b1.toString());

        b5.changePath("Spiral");
        System.out.println("D: " + b1.toString());
        System.out.println("E: " + b5.toString());
    }
}
```

(Question 1 continued on next page)

(Question 1 continued)

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

```
See the dog on the mat.  
The mat is on the floor.
```

What will the following `printFile` method print out?

```
public void printFile (){  
    try{  
        Scanner scan = new Scanner (new File("story.txt"));  
        int w = 0;  
        while ( scan.hasNext() ){  
            String str = scan.next();  
            if ( ! str.equalsIgnoreCase("the") ) {  
                System.out.println(w + " : " + str);  
                w++;  
            }  
        }  
        System.out.println("Count = " + w);  
        scan.close();  
    }  
    catch(Exception e){System.out.println("File reading failed");}  
}
```

```
0: See  
1: dog  
2: on  
3: mat.  
4: mat  
5: is  
6: on  
7: floor.  
Count = 8
```

(Question 1 continued on next page)

(Question 1 continued)

(k) [5 marks] Complete the following `sum` method. `sum` has one parameter – an array of ints – and should return the sum of the integers in the array.

```

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

```

(l) [5 marks] Sketch what the following `drawShapes` method would draw on the canvas if it were called with `drawShapes(10)`. Assume that `canvas` is a field containing a `DrawingCanvas` object. Write the top-left coordinates of each shape on your sketch.

```

public void drawShapes(int y){
    this.canvas.drawRect(100, y, 8, 8);
    if (y <= 100){
        this.drawShapes(y+20);
    }
}

```

```

| A vertical sequence of six small squares, with
| coordinates at
| (100,10), (100,30), (100,50), (100,70), (100,90), (100,110)

```

Question 2. Files

[20 marks]

Suppose the file `ttdata.txt` contains data about the lecture times of five courses, where each line contains the course code, the days of the week, the starting time, the end time, and the lecture theatre. The file is very similar to the file used in Assignment 5, but it has only five lines.

```
QUAN102 W 1600 1700 LB117
ECON333 T 1700 1800 RWW127
CHEM206 W 1300 1700 MY301
MGMT202 W 1600 1700 RWW313
OPRE454 TF 1200 1300 MY301
```

(a) [4 marks] What will `testFiles("ttdata.txt")` print to `System.out`?

```
public void testFiles (String fname) {
    try{
        Scanner fileScan = new Scanner(new File(fname));
        String d1 = fileScan.next();
        String d2 = fileScan.next();
        while (fileScan.hasNextInt()){
            System.out.println(fileScan.nextInt ());
        }
        String d3 = fileScan.nextLine();
        fileScan.close ();
    }
    catch(IOException e){System.out.println("File reading failed: "+e);}
}
```

```
| 1600
| 1700
```

(Question 2 continued on next page)

(Question 2 continued)

(b) [8 marks] Complete the following `printLateLectures` method so it reads data from a file in the format described in part (a) and only prints the information about the lectures starting after 1530. On the example file above, it should print

```
QUAN102 W 1600 1700 LB117
ECON333 T 1700 1800 RWW127
MGMT202 W 1600 1700 RWW313
```

```
public void printLateLectures(String fname){
    try{
        Scanner fileScan = new Scanner(new File(fname));
        while(fileScan.hasNext()) {
            String c = fileScan.next();
            String d = fileScan.next();
            int start = fileScan.nextInt();
            String rest = fileScan.nextLine();
            if (start > 1530){
                System.out.printf("%s %s %d %s\n", c, d, start, rest);
            }
        }
        fileScan.close();
    }
    catch(IOException e){System.out.println("File reading failed: "+e);}
}
```

(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) [8 marks] Write a method named `roomOccupancy` that has two parameters — the name of a data file (with the format described in part (a)) and the name of a lecture theatre. It should read the data from the file in the format and then return the total number of hours per week that the lecture theatre is occupied.

For example, calling `roomOccupancy("ttdata.txt", "MY301")` should *return* the value 6.

(Note: CHEM206 has 4 hours on Wednesday, and OPRE454 has one hour on Tuesday and Friday.)

You may assume that all times in the data file start on the hour.

```

public int roomOccupancy(String fname, String room){
    try{
        Scanner fileScan = new Scanner(new File(fname));
        int ans = 0;
        while(fileScan.hasNext()) {
            String crse = fileScan.next();
            String days = fileScan.next();
            int start = fileScan.nextInt();
            int end = fileScan.nextInt();
            String r = fileScan.next();
            if (room.equals(r)){
                ans = ans + (end-start)/100 *days.length();
            }
        }
        fileScan.close();
        return ans;
    }

}

}
catch(IOException e){System.out.println("File reading failed: "+e);}
}

```

Question 3. Programming with Loops

[20 marks]

The following `findDuplicates` method is intended to find duplicate words in an array of words. If each word in the array only occurs once, `findDuplicates` should print out a line saying "No duplicates". If any words occur twice in array, it should print out all the duplicate words on one line, separated by commas, and a full stop at the end of the line. Assume that no word occurs more than twice in the array.

For example, if the array contained the words:

```
"bee", "ant", "fox", "bee", "cat", "cat", "ant"
```

`printDuplicates` should print out

```
bee, ant, cat.
```

The version of `printDuplicates` below has several errors.

```
public void printDuplicates( String[ ] words){
    int count = 0;
    for ( int i=1; i<words.length; i++){           // starts from 1, not 0
        for ( int j=1; j<words.length; j++){       // starts from 1 not i+1
            if ( words[i].equals(words[j]) ){
                System.out.print(", " + words[i]); // doesn't check for first
            }                                       // to not print comma
            count++;                               // increments count every time
        }
    }
    if (count == 0) {                             // Doesn't print full stop
        System.out.println("No duplicates");       // and end of line
    }
}
```

(a) [7 marks] The version of `printDuplicates` above has errors. What does it print out if it is called with an argument that is an array containing the words

```
"bee", "ant", "fox", "bee", "cat", "cat", "ant"
```

```
| , ant, ant, fox, bee, cat, cat, cat, cat, ant, ant
```

(b) [3 marks] Circle three errors in the version of `printDuplicates` above.

Answer on the code above!

(Question 3 continued on next page)

(Question 3 continued)

(c) [10 marks] Write a correct version of `printDuplicates` so that it does what it is supposed to do.

```
public void printDuplicates( String[ ] words){
    int count = 0;
    for ( int i=0; i<words.length; i++){
        for ( int j=i+1; j<words.length; j++){
            if ( words[i].equals(words[j]) ){
                if ( count>0 ){
                    System.out.print(", ");
                }
                System.out.print(words[i]);
                count++;
            }
        }
    }
    if (count == 0) {
        System.out.println("No duplicates");
    }
    else{
        System.out.println(" . ");
    }
}
```

Question 4. Arrays of Objects

[27 marks]

This question concerns a program for keeping track of a collection of DVDs and how many times they have been viewed. The program includes a DVD class and a DVDCollection class.

The DVD class below represents information about individual DVDs.

```
public class DVD {
    private String movie;
    private int runningTime;
    private int timesViewed = 0;

    /* Constructor that takes a movie name and a running time as parameters */
    public DVD (String m, int t) {
        this.movie = m;
        this.runningTime = t;
    }
    /* Constructor that asks the user for the movie name and running time */
    public DVD () {
        Scanner s = new Scanner(System.in);
        System.out.print("Movie title: ");
        this.movie = s.nextLine();
        System.out.print("Running time: ");
        this.runningTime = s.nextInt();
    }
    /** Returns true if the DVD has the specified movie, false otherwise */
    public boolean hasName(String title) {
        return (this.movie.equals(title));
    }
    /** Returns the number of times the movie has been viewed */
    public int viewed() {
        return this.timesViewed;
    }
    /** Records that the DVD has been viewed an additional time */
    public void addViewing() {
        this.timesViewed++;
    }
    /** Returns a string description of the DVD*/
    public String toString () {
        String ans = this.movie+ " (" +this.runningTime + " mins), ";
        if (this.timesViewed == 0)
            ans = ans + "never viewed";
        else
            ans = ans + "viewed " + this.timesViewed + " times";
        return ans;
    }
}
```

(Question 4 continued on next page)

(Question 4 continued)

(a) [5 marks] The following test method tests the methods of the DVD class. What will it print out to System.out? Assume that the user answers *Antz* and *100* if the program prompts for input.

```
public void test() {
    DVD[] DVDs = new DVD[3];

    DVD b = new DVD("Shrek", 90);
    DVDs[0] = b;
    System.out.println(Dvds[0].toString ());

    DVDs[1] = new DVD("Ice Age", 15);
    DVDs[1].addViewing();
    DVDs[1].addViewing();
    System.out.println(Dvds[1].toString ());

    DVDs[2] = new DVD();

    for (int i=0; i<3; i++){
        if (DVDs[i].hasName("Shrek")){
            System.out.println(Dvds[i].viewed() + " viewings!");
        }
    }
}
```

```
Shrek (90 mins), never viewed
Ice Age (15 mins), viewed 2 times
Movie title: Antz
Running time: 100
0 viewings
```

(b) [4 marks] The DVDCollection class uses the DVD class, and contains an array to store the information about a collection of DVDs. Declare and assign initial values for fields of the DVDCollection class so that a DVDCollection object can hold information on up to 1000 DVDs.

```
public class DVDCollection{
    private static final int maxDVDs = 1000;
    private DVD[] DVDs = new DVD[maxDVDs];
    private int count;
//OR
    private DVD[] DVDs = new DVD[1000];
```

(Question 4 continued on next page)

(Question 4 continued)

(c) [8 marks] Complete the following `addNewDVD` method in the `DVDCollection` class which should allow a user to add a DVD to the collection, entering the title of the movie and the running time. If the array of DVDs is full, the method should print `The collection is full`. Your method should use the appropriate constructors and/or methods of the `DVD` class.

```
public void addNewDVD(){
    if ( this.count < this.dvds.length) {
        this.dvds[this.count]= new DVD();
        this.count++;
    }
    else{
        System.out.println("The collection is full");
    }
//OR (using array without a count field )
    for( int j=0; j<this.dvds.length; j++) {
        if (this.dvds[j] == null){
            this.dvds[j]= new DVD();
            return;
        }
    }
    System.out.println("The collection is full");
}
```

(Question 4 continued on next page)

(Question 4 continued)

(d) [10 marks] Complete the `recordViewing` method below so that it allows the user to record that they have viewed a DVD. The method should ask the user (in the terminal window) for the title of the movie, and it should then find the DVD in the collection and record another viewing of the DVD. If there is no such DVD in the collection, it should print the message `DVD not found`.

You may assume that all the DVDs have different titles.

You should use appropriate constructors and/or methods from the `DVD` class.

```
public void recordViewing(){
    System.out.print("Enter Title: ");
    Scanner scan = new Scanner(System.in);
    String title = scan.nextLine();
    for( int i= 0; i<this.count; i++) {
        if ( this.dvds[i].hasName(title) ){
            this.dvds[i].addViewing();
            return;
        }
    }
    System.out.println("DVD not found");
    //OR (using array without a count field )
    Scanner scan = new Scanner(System.in);
    String title = scan.nextLine();
    for( int i= 0; i<this.dvds.length; i++) {
        if ( this.dvds[i]!= null && this.dvds[i].hasName(title) ){
            this.dvds[i].addViewing();
            return;
        }
    }
    System.out.println("DVD not found");
}
```

Question 5. Designing with Interfaces

[30 marks]

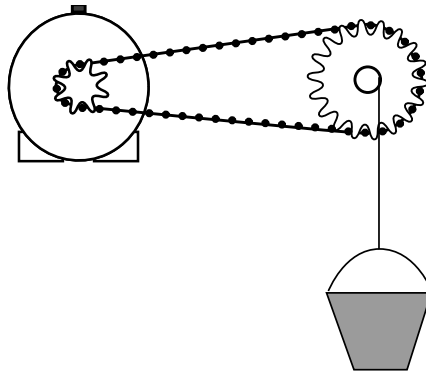
(a) [2 marks] Name two things that can be declared/defined in an ordinary class which cannot be declared in an interface class? (Note, this does not refer to user interfaces).

```
Method bodies
Constructors
Fields
```

(b) [2 marks] Suppose Chairs are one kind of Furniture, where Furniture is an *interface* defining a type. Complete the following header for the Chair class that will declare that all Chairs are Furniture:

```
public class Chair | implements Furniture {
```

The rest of this question concerns part of a program for an educational computer game which lets the user assemble simple mechanisms on the screen, made out of various components. The user can then operate the mechanism to see how it would work. For example, the user might construct a mechanism with a motor, a chain, a gear, and a bucket:



The main class of the program is called Machine. Machine has a field called components that is an array containing the different components of the current mechanism on the screen.

There are several different kinds of components, including motors, chains, gears, and buckets. You have chosen to define four classes (Motor, Chain, Gear, and Bucket) for these different kinds of components. The different components have different fields for storing information, are drawn differently, and operate in different ways. However, they all have the following three methods.

- draw, for drawing a picture of the component on a window. draw has one parameter: the DrawingCanvas to draw on.
- connect, which connects the component to another component (so that operating the component will make the other component operate. connect has one parameter - the other component.
- operate(), which makes the component turn or move and animate itself on the window. It has no parameters.

(Question 5 continued on next page)

(Question 5 continued)

(c) [7 marks] Define an interface class called **Component** to represent all categories of components, specifying the three methods that can be called on any object of type **Component**.

```
public interface Component{  
  
    public void draw(DrawingCanvas canvas);  
    public void connect(Component other);  
    public void operate();  
  
}
```

(d) [5 marks] Give an appropriate declaration for the **components** field in the **Machine** class to hold the **Motors**, **Chains**, **Gears**, **Buckets**, *etc* in the current mechanism, along with a field for the count of the items. The field declarations should also initialise the fields appropriately.

```
public class Machine{  
  
    private int MaxItems = 100;  
    private Component[ ] components = new Component[MaxItems];  
    private int count = 0;
```

(e) [5 marks] Explain why it is a good idea to define a **Component** interface for this program.

Because otherwise it would be hard to store all the different types of component in an array. We could store them all in an array of **Object**, but then it would be difficult to work out what kind of object they were when we got them out of the array.

(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)

The `Motor` class represents motor components that can drive any other component that it is connected to. When the user clicks the switch on the top of the picture of a motor, the `Machine` class will call the `operate` method on the `Motor` object. The `operate` method of the `Motor` object will first call the `operate` method on any other component to which the motor is connected, and then call the `animate` method on the motor itself.

(f) [9 marks] Complete the following header and the field declarations of the `Motor` class, and then complete the `connect` and the `operate` methods.

```

public class Motor .....implements Component {
    // Fields
    private Component connectedTo; // the component the motor is connected
    :
    :
    //Constructor
    // you do not need to define the constructor

    /* Record the component this motor is connected to */
    public void connect(      Component other ){
        this.connectedTo = other;

    }

    public void operate( ){
        if (this.connectedTo != null){
            this.connectedTo.operate();
        }
        this.animate();
    }

    /*makes the motor on the screen appear to be turning */
    private void animate(){
        // you do not need to implement this method
    }
}

```

Question 6. Recursion

[23 marks]

(a) [7 marks] The following recursive method, `sumOdd`, takes one argument, `num`, which will be an odd number, and returns the sum of all the odd numbers from `num` down to 1. For example, `sumOdd(7)` will return 16, which is $7 + 5 + 3 + 1$.

Complete the `sumOdd` method using recursion; do not use a `while` or a `for` loop.

```
public int sumOdd(int num){
    if (num == 1){
        return 1;
    }
    else {
        return ( num + sumOdd(num-2) );
    }
}
```

(Question 6 continued on next page)

(Question 6 continued)

- (b) [10 marks] What will be printed out if the `recNum` method below is called by the statement:
`System.out.println("Finally: " + recNum(15, 23));`

```
public int recNum(int x, int y){
    System.out.printf("recNum(%d, %d)\n", x, y);
    int ans;
    if (x == 0)      return 0;
    else if (y == 1) ans = x;
    else if (x > y)  ans = recNum(x-y, y);
    else            ans = recNum(y, x);

    return ans;
}
```

```
recNum(15, 23)
recNum(23, 15)
recNum(8, 15)
recNum(15, 8)
recNum(7, 8)
recNum(8, 7)
recNum(1, 7)
recNum(7, 1)
Finally: 7
```

- (c) [6 marks] Give two different sets of arguments for `recNum` above that will cause it to recurse for ever in two different ways, and explain why `recNum` will not hold in each case.

It won't stop if $x = y$ (but not $= 0$ or 1), because then `recNum` will continue to call itself (reversing its arguments) for ever.

If x is a (positive) multiple of y , then x will be reduced in the recursive calls until it is equal to y , and then it will recurse for ever.

Also, if y is negative, and x is larger than y , then the recursive call will have a larger x , so it will go on forever

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.

(You may detach this page)

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)
// (throws exception if no more tokens)
public String nextLine() // Returns the next line
// (throws exception if no more tokens)
public int nextInt() // Returns the integer value of the next token
// (throws exception if next token is not an integer
// or no more tokens)

public double nextDouble() // Returns the double value of the next token
// (throws exception if next token is not a number
// or no more tokens)
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)
```

(Continued on next page)

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 String toUpperCase(String s) // Returns upper case copy of string
public String toLowerCase(String s) // Returns lower case copy of string
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 the index of where s first matches
// Returns -1 if string does not contain s anywhere
```

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
```

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
```

MouseListener interface:

```
public void mousePressed(MouseEvent e); // Called when mouse pressed
public void mouseReleased(MouseEvent e); // Called when mouse released
public void mouseClicked(MouseEvent e); // Called when mouse clicked
public void mouseEntered(MouseEvent e); // Called when mouse enters component
public void mouseExited(MouseEvent e); // Called when mouse exits component
```

MouseEvent class:

```
public int getX() // Get the x component of the mouse position
public int getY() // Get the y component of the mouse position
```

(You may detach this page)

Small fragments of programs with examples of Java syntax

```
public void prompt(){
    Scanner sc = new Scanner(System.in);
    System.out.print("answer: ");
    if (sc.hasNextInt()){
        int n = sc.nextInt ();
        System.out.println(n);
    }
    else{
        String s = sc.next ();
        System.out.println(s);
    }
}
```

```
public void table(int n){
    for (int i=1; i<=n; i++){
        for (int j=1; j<=n; j++){
            System.out.printf(" %3d |", i*j);
        }
        System.out.println ();
    }
}
```

```
public String longest(String fname){
    String[] words = new String[20];
    try{
        Scanner f = new Scanner(new File(fname));

        int i =0;
        while (f.hasNext() && i < words.length){
            words[i] = f.next ();
            i++;
        }
    }
    catch(Exception e){System.out.println("File broken: "+ e);}

    String ans = "";
    for (int j=0; j<words.length; j++){
        if ( words[j].length() > ans.length() ){
            ans = words[j];
        }
    }
    return ans;
}
```

(Continued on next page)

```
public void actionPerformed(ActionEvent e){  
    String button = e.getActionCommand();  
    if ( button.equals("Clear") )  
        this.canvas.clear();  
    else if ( button.equals("Quit"))  
        frame.dispose();  
}
```

```
public void mouseClicked(MouseEvent e){  
    this.canvas.fillOval (e.getX(), e.getY(), 10, 10);  
}
```

```
public class Flag{  
    private int size = 40;  
    private int x;  
    private int y;  
  
    public Flag(int u, int v){  
        this.x = u;  
        this.y = v;  
    }  
  
    public void draw(DrawingCanvas canvas){  
        canvas.setColor(Color.black);  
        int left = this.x - this.size/2;  
        int top = this.y - this.size/2;  
        canvas.drawRect(left, top, this.size, this.size);  
        canvas.setColor(Color.green);  
        canvas.drawLine(this.x, top, this.x, top+this.size);  
        canvas.drawLine(left, this.y, left +this.size, this.y);  
    }  
}
```