



EXAMINATIONS — 2008

MID-YEAR

**COMP 102
INTRODUCTION TO
COMPUTER PROGRAM
DESIGN**

Time Allowed: 3 Hours ******* WITH SOLUTIONS *******

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	[61]
2. Files	[18]
3. Arrays of Objects	[35]
4. Recursion	[15]
5. Designing with Interfaces	[30]
6. Debugging Loops	[21]

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

[61 marks]

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

```

double x = 7.0;
int y = 3;
double ans = x * y;
x = x / 2;
y = y / 2;
System.out.println("x=" + x);
System.out.println("y=" + y);
System.out.println("ans=" + ans);

```

```

x=3.5
y=1
ans=21.0

```

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

```

public String compute(String s) {
    if (s.length() < 2)
        return "too small";
    else if (s.equals("four") && s.length() == 3)
        return "yes";
    else if (s.equalsIgnoreCase("x") || s.equalsIgnoreCase("yY"))
        return "XandY";
    else if (s.length() >= 5 && s.startsWith("Y"))
        return "NO";
    else
        return "all true";
}

```

What would the following calls to compute return?

```

compute("yY") ==> XandY
compute("x") ==> too small
compute("four") ==> all true
compute("Yesterday") ==> NO

```

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

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

(d) [6 marks] What will the following fragment of Java print out?

```
int n = 5;
int m = 3;
while (true){
    if (n > m)
        n = n - m;
    else
        m = m - n;

    if (n == 0 || m == 0)
        break;
    System.out.printf("Loop: n = %d, m = %d\n", n, m);
}
System.out.printf("Final: n = %d, m = %d\n", n, m);
```

```
Loop:  n = 2, m = 3
Loop:  n = 2, m = 1
Loop:  n = 1, m = 1
Final:  n = 1, m = 0
```

(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 = {"USA","CAN","JPN","DRG","PRC","GBR","NZL", "AUS"};
```

words:	USA	CAN	JPN	DRG	PRC	GBR	NZL	AUS
	0	1	2	3	4	5	6	7

What will the following code fragment print out?

```
int index=1;
while( index < words.length ){
    System.out.print(index + " : " + words[index]+ " , ");
    index = index*2;
}
```

```
1:  CAN, 2:  JPN, 4:  PRC
```

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

```
String [ ] words = {"USA","CAN","JPN","DRG","PRC","GBR","NZL", "AUS"};
```

Show the contents of `words` after the following `modify` method is called on `words`:
`modify(words)`.

```
public void modify(String [ ] wds){
    for( int i = 1; i < wds.length; i++){
        wds[i] = wds[wds.length-i];
    }
}
```

words:	USA	AUS	NZL	GBR	PRC	GBR	NZL	AUS
	0	1	2	3	4	5	6	7

(Question 1 continued on next page)

(Question 1 continued)

The `Ball` class on the facing page defines `Ball` objects, which have two fields to store their size and shape, and two methods. The class also contains a `test` method.

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

- A: 10.0 cm and Oval
- B: 3.5 cm and Oval
- C: 20.0 cm and Oval
- D: 20.0 cm and Oval
- E: 28.0 cm and Sphere

(h) [6 marks] Write an additional method called `puntAble` for the `Ball` class which returns `true` if the size of the `Ball` is above 20.0 and the shape is an `Oval`. It should return `false` in all other cases.

```
public boolean puntAble(){  
    return (this.size > 20.0 && (this.shape.equals("Oval")));  
}
```

(Question 1 continued on next page)

(Question 1 continued)

```
public class Ball{
    private double size;
    private String shape;

    public Ball(double z){
        this.size = z;
        this.shape = "Oval";
    }

    public void pumpUp(double factor){
        this.size = this.size * factor;
        if (factor > 3.0)
            this.shape = "Sphere";
    }

    public String toString(){
        return (this.size + "cm and " +this.shape);
    }

    public static void test(){
        Ball b1 = new Ball(10.0);
        Ball b4 = new Ball(3.5);

        System.out.println("A: " + b1.toString());
        System.out.println("B: " + b4.toString());

        b1.pumpUp(2.0);
        b4.pumpUp(2.0);
        System.out.println("C: " + b1.toString());

        b4.pumpUp(4.0);
        System.out.println("D: " + b1.toString());
        System.out.println("E: " + b4.toString());
    }
}
```

(Question 1 continued on next page)

(Question 1 continued)

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

```
Tell the story
tell it true
charm it crazy
[1st edition]
```

What will the following `printFile` method print out?

```
public void printFile (){
    try{
        Scanner scan = new Scanner (new File ("samhunt.txt"));
        int z = 0;
        while ( scan.hasNext() ){
            String str = scan.nextLine();
            if ( str.contains("it") ) {
                System.out.println(z + " : " + str);
                z++;
            }
            else System.out.println(" (" + str + ") ");
        }
        scan.close();
    }
    catch(Exception e){System.out.println("File reading failed");}
}
```

```
(Tell the story,)
0:tell it true
1:charm it crazy
2:[1st edition]
```

(Question 1 continued on next page)

(Question 1 continued)

(j) [6 marks] Complete the following `max` method. `max` has one parameter – an array of ints – and should return the value of the largest number in the array. You may assume that the length of the array is greater than 0.

```

public int max(int [ ] data){
    int ans = data[0];
    for (int i=1; i<data.length; i++){
        if (data[i] > data[i-1]) ans = data[i];
    }
    return ans;
//OR (alternative loop :)
    for (int n : data){ if ( n > ans ) ans = n; }
//OR (alternative start :)
    int ans = Integer.MIN_VALUE;
    for (int i=0; i<data.length; i++){ if (data[i] > data[i-1]) ans = data[i]; }
    return ans;
}

```

(k) [6 marks] Complete the following definition of a `Country` class. `Country` objects should have one field called `code` which holds a `String`. The class should have a constructor that takes one argument and sets the `code` field to its argument. The class should have one method called `getCode` (with no parameters) which returns the value of the `code` field.

```

public class Country {
    private String code;

    public Country(String c){
        this.code = c;
    }

    public String getCode(){
        return this.code;
    }

}

```

Question 2. Files

[18 marks]

Suppose the file `flightdata.txt` contains data about the daily departures of five international flights, where each line contains the flight number, the departure time, and the destination:

```
QF5 0630 Sydney
UA3 0810 Boston
BA4 0810 London
ZA9 1230 Harare
QF1 1245 London
SA6 1600 Durban
```

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

```
public void testFiles (String fname) {
    try{
        Scanner fileScan = new Scanner (new File (fname));
        while (fileScan.hasNext()){
            if (fileScan.hasNextInt()){
                int time = fileScan.nextInt ();
                String line = fileScan.nextLine ();
                System.out.println ( line );
            }
            else{
                String junk = fileScan.next ();
            }
        }
        fileScan.close ();
    }
    catch(IOException e){System.out.println("File reading failed: "+e);}
}
```

```
Sydney
Boston
London
Harare
London
Durban
```

(Question 2 continued on next page)

(Question 2 continued)

(b) [6 marks] Complete the following `printEarlyDepartures` method so it reads data from a file in the format described in part (a) and prints out the information about the planes departing before 8:30 in the morning. On the example file above, it should print

```
QF5  630  SYDNEY
UA3   810  BOSTON
BA4   810  LONDON
```

```
public void printEarlyDepartures(String fname){
    try{
        Scanner fileScan = new Scanner (new File (fname));

        while(fileScan.hasNext()) {
            String flight = fileScan.next();
            int depTime = fileScan.nextInt();
            String destination = fileScan.nextLine();
            if (depTime < 830){
                System.out.printf ("%s %d %s\n", flight, depTime, destination);
            }
        }

        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] Complete the following `firstAndLast` method which has two parameters — the name of a data file (with the format described in part (a)) and the name of a destination city. It should read the data from the file and then print the time of the earliest flight to that destination and the time of the latest flight to that destination.

For example, calling `firstAndLast("flightdata.txt", "London")` should print:

```
First 810
Last 1245
```

You may assume that there is at least one flight to the destination in the file and that the lines in the file are ordered according to departure time.

```
public void firstAndLast(String fname, String destn){
    try{
        Scanner fileScan = new Scanner(new File(fname));

        int time = -1;
        while(fileScan.hasNext()) {
            String flight = fileScan.next();
            int depTime = fileScan.nextInt();
            String d = fileScan.next();
            if (d.equalsIgnoreCase(destn)) {
                if (time < 0)
                    System.out.println("First: "+depTime);
                time = depTime;
            }
        }
        System.out.println("Last: " + time);

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

Question 3. Arrays of Objects

[35 marks]

This question concerns a program for keeping track of the current repair jobs for a computer shop. Jobs are added when a customer brings in a computer to be repaired and deleted when the computer is returned to the customer. The program keeps track of the current status of jobs (waiting, being worked on, or finished). The status of a job can be updated and the program can list the current jobs of any status. The program includes a `Job` class and a `JobTracker` class.

The `Job` class below represents information about individual Jobs.

```
public class Job{
    // Fields
    private String customer;
    private String fault ;
    private String status;    // "waiting" "repairing" or "finished"

    /** Construct a new Job object */
    public Job(String name, String fault){
        this.customer = name;
        this.fault = fault ;
        this.status = "waiting";
    }

    /** Returns a String describing the job */
    public String description(){
        return ("Fixing "+ this.fault + " for " + this.customer +
            ", status: " + this.status );
    }

    /** Returns true if and only if the job's customer is this name */
    public boolean hasName(String name){
        return (this.customer.equalsIgnoreCase(name));
    }

    /** Returns the current status of the job */
    public String getStatus(){
        return this.status;
    }

    /** Sets the current status of the job to "finished" */
    public void finish(){
        this.status = "finished";
    }
}
```

(Question 3 continued)

(a) [5 marks] The following test method tests the methods of the Job class. What will it print out to System.out?

```
public static void test () {

    Job job = new Job("Peter", "Broken Monitor");
    System.out.println(job.description ());

    String oldStatus = job.getStatus ();
    job.finish ();
    System.out.println("Job: " + oldStatus + "->" + job.getStatus());

    Job[ ] allJobs = new Job[5];
    allJobs[0] = job;
    allJobs[1] = new Job("Marcus", "No power supply");

    for (int i=0; i<2; i++){
        if (allJobs[i].hasName("Peter"))
            System.out.println("Job " + i+ " is for Peter");
        else
            System.out.println("Job " + i+ " is not for Peter");
    }
}
```

```
Fixing Broken Monitor for Peter, status:  waiting
Job:  waiting->finished
Job 0 is for peter
Job 1 is not for peter
```

(b) [4 marks] The JobTracker class uses the Job class, and contains an array to store the information about a collection of Jobs. Declare and assign initial values for fields of the JobTracker class so that a JobTracker object could hold information on up to 100 Jobs. It should use an array and a count.

```
public class JobTracker{
    private final int maxJobs = 100;
    private Job[] jobs = new Job[maxJobs];
    private int count = 0;
```

(Question 3 continued on next page)

(Question 3 continued)

(c) [6 marks] Complete the following `listWaitingJobs` method in the `JobTracker` class which should print (to `System.out`) a description of each job that has a status of `"waiting"`. Your method should use the appropriate methods of the `Job` class.

```
public void listWaitingJobs(){  
    for (int i=0; i<this.count; i++){  
        if (this.jobs[i].getStatus().equals("waiting")){  
            System.out.println(this.jobs[i].description ());  
        }  
    }  
  
}
```

(Question 3 continued on next page)

(Question 3 continued)

(d) [8 marks] Complete the following `addJob` method in the `JobTracker` class which should allow a user to add a new `Job` to the collection. If the array of jobs is full, the method should print `The collection is full` and should not ask the user for any data. Otherwise, the method should ask the user (in the terminal window) for the customer name and a description of the fault, and then add the new job to the collection.

Your method should use the appropriate constructors and/or methods of the `Job` class.

```
public void addJob(){
    if (this.count >= this.jobs.length){
        System.out.println("The collection is full");
        return;
    }
    Scanner sc = new Scanner (System.in);
    System.out.print("Customer name: ");
    String name = sc.nextLine();
    System.out.print("Job description: ");
    String descr = sc.nextLine();

    this.jobs[this.count] = new Job(name, descr);
    this.count++;

}
```

(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)

(e) [8 marks] Complete the following `finishJob` method in the `JobTracker` class so that it allows the user to update the status of a job to `"finished"`. The parameter of the method is the customer name. It should find the job with that name, update its status, and then print `"Updated status"`. If there is no job with the given name, it should print `"No such job"`. You should use appropriate methods from the `Job` class. You may assume there is only one job for each customer.

```

public void finishJob( String name){
    for ( int i=0; i<this.count; i++){
        if (this.jobs[i].hasName(name)){
            this.jobs[i].finish ();
            System.out.println("Updated status");
            return;
        }
    }
    System.out.println("No such job");
}
}

```

(f) [4 marks] There is a problem with the design of the program, especially the `finishJob` method, if there can be more than one current repair job for the same customer. Explain the problem and suggest how you would change the design to allow a customer to have multiple repair jobs.

If there are two repair jobs for the same person, the `finishJob` method will only find the first one, and will update it, even if it was the other job that was finished. We need more details than just the name to identify a job. There are many possible solutions:

- give a unique job id to every job,
- use the name and the fault to identify the job
- find (and display) all jobs with the given name and ask the user to select the one they want.
- put a barcode on the computer and read it with a barcode scanner to identify the job.

Allow all of these options, with a search interface that allows retrieving by any of the fields.

Question 4. Recursion

[15 marks]

(a) [7 marks] The following recursive method, `recDraw`, draws rectangles on the canvas. In the box below, sketch what it would draw if it were called by the statement:

```
this.recDraw(30, 10, 4);
```

Label the rectangles with their position and size.

```
public void recDraw(int m, int p, int num){  
    if (num > 0){  
        this.canvas.drawRect(m, m, p, p);  
        this.recDraw(m+p, p*2, num-1);  
    }  
}
```

```
Rectangle at (30, 30), size 10  
rectangle at (40, 40), size 20  
rectangle at (60, 60), size 40  
rectangle at (100, 100), size 80
```

(Question 4 continued on next page)

(Question 4 continued)

(b) [8 marks] The following recursive method, `printRec`, is passed a `Scanner` that is already connected to a file. The method should read the lines in the file, and print out all the lines in order, followed by all the lines in reverse order. Complete the definition of `printRec`.

You *must* use recursion: do *not* use loops or arrays.

For example, if the file contains the text shown on the left, the method should print out the lines shown on the right.

```
This first line is
before the 2nd
and the third.
```

```
This first line is
before the 2nd
and the third.
and the third.
before the 2nd
This first line is
```

```
public void printRec(Scanner scan){
    if (scan.hasNext()){
        String line = scan.nextLine();
        System.out.println(line);
        printRec(scan);
        System.out.println(line);
    }
}
```

Question 5. Designing with Interfaces

[30 marks]

Suppose the interface class `Item` is defined as below, and that `Cornflakes` is a class that implements `Item`.

```
public interface Item {
    public double price ();
    public void writeEntry( int count);
}

public class Cornflakes implements Item {
    :
}
```

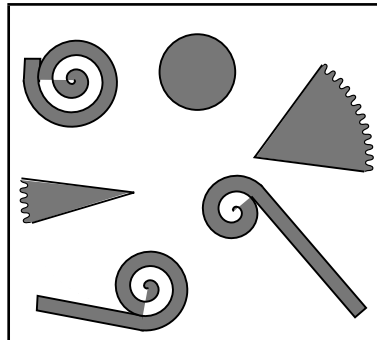
(a) [4 marks] For each of the following assignment statements, state whether it is valid or invalid. If it is invalid, explain why.

<code>Item it = new Item();</code>	//Ans: INVALID: Item has no constructor.
<code>Item itm = new Cornflakes();</code>	//Ans: valid
<code>double pr2 = itm.price ();</code>	//Ans: valid
<code>Object ob = new Cornflakes();</code>	//Ans: valid
<code>double pr3 = ob.price ();</code>	//Ans: INVALID: Object has no price method

(Question 5 continued on next page)

(Question 5 continued)

The rest of this question concerns a `BlowUpGame` program that is an extended version of the `BalloonGame` from assignment 8. `BalloonGame` displayed a collection of `Balloon` objects on the window, allowed the user to expand them, and popped balloons that touched. `BlowUpGame` has a collection of three different kinds of expandable items — balloons, party whistles, and bellows. The figure below shows a collection with three party whistles, two bellows, and one balloon.



The main class of the program is `BlowUpGame`. It has a field called `items` that contains an array with all the current expandable items.

The program also contains three classes (`Balloon`, `PartyWhistle`, and `Bellows`) for the different kinds of expandable items. Each class has different fields for storing information, and the objects have different shapes, and expand differently. However, all three classes have the following four methods.

- `draw`, which has one `DrawingCanvas` parameter. It draws the item on the canvas.
- `pointOn`, which has two integer parameters specifying a point. It returns true if and only if the point is on the item.
- `expand`, which has one integer parameter. It makes the item larger in the appropriate way. (Balloons just get bigger, party whistles unroll, and bellows open up.)
- `touching`, which has one parameter that is another expandable item. It returns true if and only if this item is touching the other item.

(b) [6 marks] Define an interface class called `ExpandableItem` to represent all three types of expandable items, specifying the four methods that can be called on any object of type `ExpandableItem`.

```
public interface ExpandableItem{

    public void draw(DrawingCanvas canvas);
    public boolean pointOn(int x, int y);
    public void expand(int step);
    public boolean touching (ExpandableItem other);

}
```

(c) [3 marks] Give an appropriate declaration for the items field in the BlowUpGame class to hold up to 100 Balloons, PartyWhistles, and Bellows. The field declaration should also initialise the field appropriately.

```
public class BlowUpGame{  
  
    private ExpandableItem[] items = new ExpandableItem[100];  
  
    : // you do not need to declare the other fields .
```

(d) [4 marks] Explain why it is a good idea to define an ExpandableItem interface for this program.

Because otherwise it would be hard to store all the different types of items 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.

The PartyWhistle class represents party whistle items which are initially rolled up, and can unroll to a maximum length of 200. At any point in time, a PartyWhistle object will be rolled out by some fraction of its maximum length. Each PartyWhistle object also has a position and the direction that the entrance is facing (a double), both of which are set when the object is created.

(e) [8 marks] Complete the following header, field declarations, and constructor of the PartyWhistle class.

```
public class PartyWhistle implements ExpandableItem {  
    // Fields  
    private int x;  
    private int y;  
    private double direction;  
    private int maxLength = 200; // not strictly required, but good practice  
    private double rollout = 0.0; // or could be an int  
  
    //Constructor  
    public PartyWhistle(int x, int y, int dir){ // could have additional parameters.  
        this.x = x;  
        this.y = y;  
        this.direction = dir;  
    }  
  
    // You do not need to define the methods  
}
```

(Question 5 continued on next page)

(Question 5 continued)

(f) [5 marks] Implementing the touching method for the Balloon class in the BalloonGame program was relatively easy. Give at least two different reasons why it would be more difficult to implement the touching methods for the three different expandable item classes.

- (a) Because the shapes are more complicated, so the geometry would be more tricky
- (b) Because the touching method in the Balloon class would need to know what kind of item the other item was - it would need to do a different computation if the other item was a Balloon or a PartyWhistle or a Bellows. The same is true for the other two classes
- (c) Because the touching methods in each class could not access the the fields of the other object if it were from a different class

??? Explain how you would redesign it?

Question 6. Debugging Loops

[21 marks]

The following `listIdenticalRows` method is intended to find pairs of identical rows in a table of numbers stored in a 2D array. It should print out the row numbers of each pair of identical rows, and should print the total number of pairs of rows at the end. The rows are numbered from 0.

For example, for each array of numbers on the left, `listIdenticalRows` should print out the text on the right.

	0	1	2		
0	1	5	21	⇒	same: 1 & 3 number of pairs: 1
1	3	6	14		
2	3	2	21		
3	3	6	14		
	0	1	2		
0	1	7	21	⇒	same: 0 & 2 same: 0 & 4 same: 1 & 3 same: 2 & 4 number of pairs: 4
1	3	6	10		
2	1	7	21		
3	3	6	10		
4	1	7	21		

The version of `listIdenticalRows` below has several errors.

```
public void listIdenticalRows (int[ ][ ] data){
    int rows = data.length;
    int cols = data[0].length;
    int count = 0;

    for (int row= 0; row<rows; row++){
        for (int otherRow= 0; otherRow<rows; otherRow++){ // (1) will count i,j and j, i
            if (otherRow != row){

                // test if rows are the same
                boolean same = false; // (2) wrong (inverted) logic
                for (int col= 0; col<cols; col++){
                    if (data[row][col] == data[otherRow][col])
                        same = true; // (2) wrong logic
                }
                if (same){
                    System.out.println("same: " + row + " & " + otherRow);
                }
                count++; // (3) counts every row
            }
        }
    }
    System.out.println("number of pairs: " + count);
}
```

(a) [8 marks] The version of `listIdenticalRows` on the facing page has errors. What does it print out if it is called with an argument that is the following 2D array of numbers (same as first example on facing page):

	0	1	2
0	1	5	21
1	3	6	14
2	3	2	21
3	3	6	14

```
same: 0 & 2
same: 1 & 2
same: 1 & 3
same: 2 & 0
same: 2 & 1
same: 2 & 3
same: 3 & 1
same: 3 & 2
number of pairs: 12
```

(b) [3 marks] Circle and briefly describe three errors in the version of `listIdenticalRows` on the facing page.

Answer on the code on the facing page

[part (c) is 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 6 continued)

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

```

public void listIdenticalRows (int [ ] [ ] data){
    int rows = data.length;
    int cols = data[0].length;
    int count = 0;

    for (int row= 0; row<rows; row++){
        for (int otherRow= row+1; otherRow<rows; otherRow++){

            boolean same = true;
            for (int col= 0; col<cols; col++){
                if (data[row][col] != data[otherRow][col])
                    same = false;
            }
            if (same) {
                System.out.println("same: " + row + " & " + otherRow);
                count++;
            }
        }
    }
    // OR
    for (int row= 0; row<rows; row++){
        for (int otherRow= 0; otherRow<rows; otherRow++){
            if (otherRow > row){
                int countSame = 0;
                for (int col= 0; col<cols; col++){
                    if (data[row][col] == data[otherRow][col])
                        countSame++;
                }
                if (countSame == cols) {
                    System.out.println("same: " + row + " & " + otherRow);
                    count++;
                }
            }
        }
    }

    System.out.println("number of pairs: " + count);
}

```

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 string of chars up to next newline
// (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
public boolean exists() // Returns true iff the file exists

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);  
    }  
}
```

```
public void print2Darray(int [][] table){  
    for ( int row=0; row<table.length; row++){  
        for ( int col=0; col<table[row].length; col++){  
            System.out.printf ("%4d", table[row][col ]);  
        }  
        System.out.println ();  
    }  
}
```
